

libcaca
0.99.beta13b

Generated by Doxygen 1.5.6

Wed Feb 25 18:25:13 2009

Contents

1	libcaca Documentation	1
1.1	Introduction	1
1.2	Developer's documentation	2
1.3	User's documentation	2
1.4	Misc	2
1.5	License	2
2	Module Documentation	3
2.1	libcucul attribute definitions	3
2.1.1	Detailed Description	3
2.1.2	Define Documentation	3
2.2	libcucul basic functions	5
2.2.1	Detailed Description	6
2.2.2	Function Documentation	6
2.3	libcucul canvas drawing	10
2.3.1	Detailed Description	11
2.3.2	Define Documentation	11
2.3.3	Function Documentation	11
2.4	libcucul canvas transformation	18
2.4.1	Detailed Description	19
2.4.2	Function Documentation	19
2.5	libcucul attribute conversions	22
2.5.1	Detailed Description	22
2.5.2	Function Documentation	22
2.6	libcucul character set conversions	25
2.6.1	Detailed Description	25
2.6.2	Function Documentation	25
2.7	libcucul primitives drawing	27
2.7.1	Detailed Description	28
2.7.2	Function Documentation	28
2.8	libcucul canvas frame handling	34
2.8.1	Detailed Description	34
2.8.2	Function Documentation	34
2.9	libcucul bitmap dithering	37
2.9.1	Detailed Description	38

2.9.2	Function Documentation	38
2.10	libcucul font handling	46
2.10.1	Detailed Description	46
2.10.2	Function Documentation	46
2.11	libcucul importers/exporters from/to various	49
2.11.1	Detailed Description	49
2.11.2	Function Documentation	49
2.12	libcaca basic functions	52
2.12.1	Detailed Description	53
2.12.2	Function Documentation	53
2.13	libcaca event handling	56
2.13.1	Detailed Description	57
2.13.2	Function Documentation	57
3	Data Structure Documentation	61
3.1	caca_event Struct Reference	61
3.1.1	Detailed Description	61
4	File Documentation	61
4.1	caca.h File Reference	61
4.1.1	Detailed Description	64
4.1.2	Define Documentation	64
4.1.3	Typedef Documentation	65
4.1.4	Enumeration Type Documentation	65
4.2	cucul.h File Reference	67
4.2.1	Detailed Description	74
4.2.2	Define Documentation	74
4.2.3	Typedef Documentation	74

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 slang library or the ncurses library, on DOS using the conio library, and on Windows systems using either slang or ncurses (through Cygwin emulation) or conio. 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 Heck 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

libcaca relies on a low-level, device independent library, called *libcucul*. *libcucul* can be used alone as a simple ASCII and/or Unicode compositing canvas.

The complete *libcucul* and *libcaca* programming interface is available from the following headers:

- [cucul.h](#)
- [caca.h](#)

There is language-specific documentation for the various bindings:

- [Libcaca ruby bindings](#)

Some other topics are covered by specific sections:

- [A libcucul and 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](#)
- [Libcaca TODO list](#)

1.5 License

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

2 Module Documentation

2.1 libcucul attribute definitions

Defines

- `#define CUCUL_BLACK 0x00`
- `#define CUCUL_BLUE 0x01`
- `#define CUCUL_GREEN 0x02`
- `#define CUCUL_CYAN 0x03`
- `#define CUCUL_RED 0x04`
- `#define CUCUL_MAGENTA 0x05`
- `#define CUCUL_BROWN 0x06`
- `#define CUCUL_LIGHTGRAY 0x07`
- `#define CUCUL_DARKGRAY 0x08`
- `#define CUCUL_LIGHTBLUE 0x09`
- `#define CUCUL_LIGHTGREEN 0x0a`
- `#define CUCUL_LIGHTCYAN 0x0b`
- `#define CUCUL_LIGHTRED 0x0c`
- `#define CUCUL_LIGHTMAGENTA 0x0d`
- `#define CUCUL_YELLOW 0x0e`
- `#define CUCUL_WHITE 0x0f`
- `#define CUCUL_DEFAULT 0x10`
- `#define CUCUL_TRANSPARENT 0x20`
- `#define CUCUL_BOLD 0x01`
- `#define CUCUL_ITALICS 0x02`
- `#define CUCUL_UNDERLINE 0x04`
- `#define CUCUL_BLINK 0x08`

2.1.1 Detailed Description

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

2.1.2 Define Documentation

2.1.2.1 `#define CUCUL_BLACK 0x00`

The colour index for black.

Referenced by `cucul_attr_to_ansi()`, `cucul_attr_to_argb64()`, `cucul_attr_to_rgb12_bg()`, and `cucul_dither_bitmap()`.

2.1.2.2 `#define CUCUL_BLUE 0x01`

The colour index for blue.

2.1.2.3 `#define CUCUL_GREEN 0x02`

The colour index for green.

2.1.2.4 #define CUCUL_CYAN 0x03

The colour index for cyan.

2.1.2.5 #define CUCUL_RED 0x04

The colour index for red.

2.1.2.6 #define CUCUL_MAGENTA 0x05

The colour index for magenta.

2.1.2.7 #define CUCUL_BROWN 0x06

The colour index for brown.

2.1.2.8 #define CUCUL_LIGHTGRAY 0x07

The colour index for light gray.

Referenced by `cucul_attr_to_ansi()`, `cucul_attr_to_argb64()`, and `cucul_attr_to_rgb12_fg()`.

2.1.2.9 #define CUCUL_DARKGRAY 0x08

The colour index for dark gray.

2.1.2.10 #define CUCUL_LIGHTBLUE 0x09

The colour index for blue.

2.1.2.11 #define CUCUL_LIGHTGREEN 0x0a

The colour index for light green.

2.1.2.12 #define CUCUL_LIGHTCYAN 0x0b

The colour index for light cyan.

2.1.2.13 #define CUCUL_LIGHTRED 0x0c

The colour index for light red.

2.1.2.14 #define CUCUL_LIGHTMAGENTA 0x0d

The colour index for light magenta.

2.1.2.15 #define CUCUL_YELLOW 0x0e

The colour index for yellow.

2.1.2.16 #define CUCUL_WHITE 0x0f

The colour index for white.

2.1.2.17 #define CUCUL_DEFAULT 0x10

The output driver's default colour.

Referenced by `cucul_attr_to_argb64()`, `cucul_attr_to_rgb12_bg()`, `cucul_attr_to_rgb12_fg()`, and `cucul_create_canvas()`.

2.1.2.18 #define CUCUL_TRANSPARENT 0x20

The transparent colour.

Referenced by `cucul_attr_to_argb64()`, `cucul_attr_to_rgb12_bg()`, `cucul_attr_to_rgb12_fg()`, and `cucul_create_canvas()`.

2.1.2.19 #define CUCUL_BOLD 0x01

The style mask for bold.

2.1.2.20 #define CUCUL_ITALICS 0x02

The style mask for italics.

2.1.2.21 #define CUCUL_UNDERLINE 0x04

The style mask for underline.

2.1.2.22 #define CUCUL_BLINK 0x08

The style mask for blink.

2.2 libcucul basic functions**Functions**

- `cucul_canvas_t * cucul_create_canvas` (unsigned int, unsigned int)
Initialise a libcucul canvas.
- `int cucul_manage_canvas` (`cucul_canvas_t *`, `int(*) (void *)`, `void *`)
Manage a canvas.
- `int cucul_unmanage_canvas` (`cucul_canvas_t *`, `int(*) (void *)`, `void *`)
Unmanage a canvas.
- `int cucul_set_canvas_size` (`cucul_canvas_t *`, unsigned int, unsigned int)
Resize a canvas.
- unsigned int `cucul_get_canvas_width` (`cucul_canvas_t const *`)
Get the canvas width.
- unsigned int `cucul_get_canvas_height` (`cucul_canvas_t const *`)
Get the canvas height.

- unsigned char const * [cucul_get_canvas_chars](#) (cucul_canvas_t const *)
Get the canvas character array.
- unsigned char const * [cucul_get_canvas_attrs](#) (cucul_canvas_t const *)
Get the canvas attribute array.
- int [cucul_free_canvas](#) (cucul_canvas_t *)
Uninitialise libcucul.
- int [cucul_rand](#) (int, int)
Generate a random integer within a range.
- char const * [cucul_get_version](#) (void)
Return the libcucul version.

2.2.1 Detailed Description

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

2.2.2 Function Documentation

2.2.2.1 cucul_canvas_t* cucul_create_canvas (unsigned int *width*, unsigned int *height*)

Initialise internal *libcucul* structures and the backend that will be used for subsequent graphical operations. It must be the first *libcucul* function to be called in a function. [cucul_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:

- ENOMEM Not enough memory for the requested canvas size.

Parameters:

width The desired canvas width

height The desired canvas height

Returns:

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

References CUCUL_DEFAULT, cucul_set_color_ansi(), and CUCUL_TRANSPARENT.

Referenced by caca_create_display(), and cucul_set_canvas_boundaries().

2.2.2.2 int cucul_manage_canvas (cucul_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 *cucul_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.

Parameters:

cv A libcucul canvas.

callback An optional callback function pointer.

p The argument to be passed to *callback*.

Returns:

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

Referenced by `caca_create_display()`.

2.2.2.3 `int cucul_unmanage_canvas (cucul_canvas_t * cv, int (*)(void *) callback, void * p)`

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

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

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

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

Parameters:

cv A libcucul canvas.

callback The *callback* argument previously passed to `cucul_manage_canvas()`.

p The *p* argument previously passed to `cucul_manage_canvas()`.

Returns:

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

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

2.2.2.4 `int cucul_set_canvas_size (cucul_canvas_t * cv, unsigned int width, unsigned int height)`

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

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

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

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

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 resized.
- **ENOMEM** Not enough memory for the requested canvas size. If this happens, the canvas handle becomes invalid and should not be used.

Parameters:

cv A libcucul canvas.

width The desired canvas width.

height The desired canvas height.

Returns:

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

2.2.2.5 unsigned int cucul_get_canvas_width (cucul_canvas_t const * cv)

Return the current canvas' width, in character cells.

This function never fails.

Parameters:

cv A libcucul canvas.

Returns:

The canvas width.

Referenced by `caca_create_display()`, and `caca_get_mouse_x()`.

2.2.2.6 unsigned int cucul_get_canvas_height (cucul_canvas_t const * cv)

Returns the current canvas' height, in character cells.

This function never fails.

Parameters:

cv A libcucul canvas.

Returns:

The canvas height.

Referenced by `caca_create_display()`, and `caca_get_mouse_y()`.

2.2.2.7 unsigned char const* cucul_get_canvas_chars (cucul_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 [cucul_get_char\(\)](#).

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

This function never fails.

Parameters:

cv A libcucul canvas.

Returns:

The canvas character array.

2.2.2.8 unsigned char const* cucul_get_canvas_attrs (cucul_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 [cucul_get_attr\(\)](#).

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

This function never fails.

Parameters:

cv A libcucul canvas.

Returns:

The canvas attribute array.

2.2.2.9 int cucul_free_canvas (cucul_canvas_t * cv)

Free all resources allocated by [cucul_create_canvas\(\)](#). After this function has been called, no other *libcucul* functions may be used unless a new call to [cucul_create_canvas\(\)](#) is done.

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

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

Parameters:

cv A libcucul canvas.

Returns:

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

Referenced by [caca_create_display\(\)](#), and [caca_free_display\(\)](#).

2.2.2.10 int cucul_rand (int min, int max)

Generate a random integer within the given range.

This function never fails.

Parameters:

min The lower bound of the integer range.

max The upper bound of the integer range.

Returns:

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

2.2.2.11 char const* cucul_get_version (void)

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

This function never fails.

Returns:

The *libcucul* version information.

2.3 libcucul canvas drawing

Defines

- #define CUCUL_MAGIC_FULLWIDTH 0x000ffffe

Functions

- int cucul_gotoxy (cucul_canvas_t *, int, int)
Set cursor position.
- int cucul_get_cursor_x (cucul_canvas_t const *)
Get X cursor position.
- int cucul_get_cursor_y (cucul_canvas_t const *)
Get Y cursor position.
- int cucul_put_char (cucul_canvas_t *, int, int, unsigned long int)
Print an ASCII or Unicode character.
- unsigned long int cucul_get_char (cucul_canvas_t const *, int, int)
Get the Unicode character at the given coordinates.
- int cucul_put_str (cucul_canvas_t *, int, int, char const *)
Print a string.
- unsigned long int cucul_get_attr (cucul_canvas_t const *, int, int)
Get the text attribute at the given coordinates.
- int cucul_set_attr (cucul_canvas_t *, unsigned long int)
Set the default character attribute.
- int cucul_put_attr (cucul_canvas_t *, int, int, unsigned long int)
Set the character attribute at the given coordinates.
- int cucul_set_color_ansi (cucul_canvas_t *, unsigned char, unsigned char)
Set the default colour pair for text (ANSI version).
- int cucul_set_color_argb (cucul_canvas_t *, unsigned int, unsigned int)
Set the default colour pair for text (truecolor version).

- `int cucul_printf (cucul_canvas_t *, int, int, char const *,...)`
Print a formatted string.
- `int cucul_clear_canvas (cucul_canvas_t *)`
Clear the canvas.
- `int cucul_set_canvas_handle (cucul_canvas_t *, int, int)`
Set cursor handle.
- `int cucul_get_canvas_handle_x (cucul_canvas_t const *)`
Get X handle position.
- `int cucul_get_canvas_handle_y (cucul_canvas_t const *)`
Get Y handle position.
- `int cucul_blit (cucul_canvas_t *, int, int, cucul_canvas_t const *, cucul_canvas_t const *)`
Blit a canvas onto another one.
- `int cucul_set_canvas_boundaries (cucul_canvas_t *, int, int, unsigned int, unsigned int)`
Set a canvas' new boundaries.

2.3.1 Detailed Description

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

2.3.2 Define Documentation

2.3.2.1 `#define CUCUL_MAGIC_FULLWIDTH 0x000ffffe`

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

Referenced by `cucul_blit()`, `cucul_flip()`, `cucul_put_attr()`, `cucul_put_char()`, and `cucul_rotate_180()`.

2.3.3 Function Documentation

2.3.3.1 `int cucul_gotoxy (cucul_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.

Parameters:

- `cv` A handle to the libcucul canvas.
- `x` X cursor coordinate.
- `y` Y cursor coordinate.

Returns:

This function always returns 0.

2.3.3.2 int cucul_get_cursor_x (cucul_canvas_t const * *cv*)

Retrieve the X coordinate of the cursor's position.

This function never fails.

Parameters:

cv A handle to the libcucul canvas.

Returns:

The cursor's X coordinate.

2.3.3.3 int cucul_get_cursor_y (cucul_canvas_t const * *cv*)

Retrieve the Y coordinate of the cursor's position.

This function never fails.

Parameters:

cv A handle to the libcucul canvas.

Returns:

The cursor's Y coordinate.

2.3.3.4 int cucul_put_char (cucul_canvas_t * *cv*, int *x*, int *y*, unsigned long int *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 [cucul_put_str\(\)](#) function.

This function never fails.

Parameters:

cv A handle to the libcucul canvas.

x X coordinate.

y Y coordinate.

ch The character to print.

Returns:

This function always returns 0.

References CUCUL_MAGIC_FULLWIDTH, and cucul_utf32_is_fullwidth().

Referenced by cucul_dither_bitmap(), cucul_draw_cp437_box(), cucul_draw_thin_box(), cucul_fill_box(), cucul_fill_triangle(), and cucul_put_str().

2.3.3.5 unsigned long int cucul_get_char (cucul_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 CUCUL_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.

Parameters:

- cv* A handle to the libcucul canvas.
- x* X coordinate.
- y* Y coordinate.

Returns:

This function always returns 0.

2.3.3.6 int cucul_put_str (cucul_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 [cucul_put_char\(\)](#) for more information on how fullwidth characters are handled when overwriting each other or at the canvas' boundaries.

This function never fails.

Parameters:

- cv* A handle to the libcucul canvas.
- x* X coordinate.
- y* Y coordinate.
- s* The string to print.

Returns:

This function always returns 0.

References [cucul_put_char\(\)](#), [cucul_utf32_is_fullwidth\(\)](#), and [cucul_utf8_to_utf32\(\)](#).

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

2.3.3.7 unsigned long int cucul_get_attr (cucul_canvas_t const * cv, int x, int y)

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

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

- 4 bits for the background green component
- 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.

Parameters:

- cv* A handle to the libcucul canvas.
- x* X coordinate.
- y* Y coordinate.

Returns:

The requested attribute.

Referenced by `cucul_dither_bitmap()`.

2.3.3.8 `int cucul_set_attr (cucul_canvas_t * cv, unsigned long int 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 `cucul_get_attr()`, in which case it also contains colour information,
- a combination (bitwise OR) of style values (`CUCUL_UNDERLINE`, `CUCUL_BLINK`, `CUCUL_BOLD` and `CUCUL_ITALICS`), in which case setting the attribute does not modify the current colour information.

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

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

- `EINVAL` The attribute value is out of the 32-bit range.

Parameters:

- cv* A handle to the libcucul canvas.
- attr* The requested attribute value.

Returns:

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

Referenced by `cucul_dither_bitmap()`.

2.3.3.9 int cucul_put_attr (cucul_canvas_t * *cv*, int *x*, int *y*, unsigned long int *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 `cucul_get_attr()`, in which case it also contains colour information,
- a combination (bitwise OR) of style values (`CUCUL_UNDERLINE`, `CUCUL_BLINK`, `CUCUL_BOLD` and `CUCUL_ITALICS`), in which case setting the attribute does not modify the current colour information.

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

- `EINVAL` The attribute value is out of the 32-bit range.

Parameters:

- cv* A handle to the libcucul canvas.
- x* X coordinate.
- y* Y coordinate.
- attr* The requested attribute value.

Returns:

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

References `CUCUL_MAGIC_FULLWIDTH`.

2.3.3.10 int cucul_set_color_ansi (cucul_canvas_t * *cv*, unsigned char *fg*, unsigned char *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 `cucul.h`, such as `CUCUL_RED` or `CUCUL_TRANSPARENT`.

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

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

Parameters:

- cv* A handle to the libcucul canvas.
- fg* The requested ANSI foreground colour.
- bg* The requested ANSI background colour.

Returns:

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

Referenced by `cucul_create_canvas()`, and `cucul_dither_bitmap()`.

2.3.3.11 int cucul_set_color_argb (cucul_canvas_t * *cv*, unsigned int *fg*, unsigned int *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).

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

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

Parameters:

- cv* A handle to the libcucul canvas.
- fg* The requested ARGB foreground colour.
- bg* The requested ARGB background colour.

Returns:

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

2.3.3.12 int cucul_printf (cucul_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 never fails.

Parameters:

- cv* A handle to the libcucul canvas.
- x* X coordinate.
- y* Y coordinate.
- format* The format string to print.
- ... Arguments to the format string.

Returns:

- This function always returns 0.

References `cucul_put_str()`.

2.3.3.13 int cucul_clear_canvas (cucul_canvas_t * *cv*)

Clear the canvas using the current foreground and background colours.

This function never fails.

Parameters:

- cv* The canvas to clear.

Returns:

- This function always returns 0.

2.3.3.14 int cucul_set_canvas_handle (cucul_canvas_t * *cv*, int *x*, int *y*)

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

This function never fails.

Parameters:

- cv* A handle to the libcucul canvas.
- x* X handle coordinate.
- y* Y handle coordinate.

Returns:

This function always returns 0.

2.3.3.15 int cucul_get_canvas_handle_x (cucul_canvas_t const * *cv*)

Retrieve the X coordinate of the canvas' handle.

This function never fails.

Parameters:

- cv* A handle to the libcucul canvas.

Returns:

The canvas' handle's X coordinate.

2.3.3.16 int cucul_get_canvas_handle_y (cucul_canvas_t const * *cv*)

Retrieve the Y coordinate of the canvas' handle.

This function never fails.

Parameters:

- cv* A handle to the libcucul canvas.

Returns:

The canvas' handle's Y coordinate.

2.3.3.17 int cucul_blit (cucul_canvas_t * *dst*, int *x*, int *y*, cucul_canvas_t const * *src*, cucul_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.

Parameters:

- dst* The destination canvas.

x X coordinate.
y Y coordinate.
src The source canvas.
mask The mask canvas.

Returns:

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

References CUCUL_MAGIC_FULLWIDTH.

Referenced by cucul_set_canvas_boundaries().

2.3.3.18 int cucul_set_canvas_boundaries (cucul_canvas_t * *cv*, int *x*, int *y*, unsigned int *w*, unsigned 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:

- **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.

Parameters:

cv The canvas to crop.
x X coordinate of the top-left corner.
y Y coordinate of the top-left corner.
w The width of the cropped area.
h The height of the cropped area.

Returns:

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

References cucul_blit(), cucul_create_canvas(), cucul_create_frame(), cucul_get_frame_count(), and cucul_set_frame().

2.4 libcucul canvas transformation

Functions

- int [cucul_invert](#) (cucul_canvas_t *)
Invert a canvas' colours.
- int [cucul_flip](#) (cucul_canvas_t *)
Flip a canvas horizontally.
- int [cucul_flop](#) (cucul_canvas_t *)

Flip a canvas vertically.

- `int cucul_rotate_180 (cucul_canvas_t *)`
Rotate a canvas.
- `int cucul_rotate_left (cucul_canvas_t *)`
Rotate a canvas, 90 degrees counterclockwise.
- `int cucul_rotate_right (cucul_canvas_t *)`
Rotate a canvas, 90 degrees counterclockwise.
- `int cucul_stretch_left (cucul_canvas_t *)`
Rotate and stretch a canvas, 90 degrees counterclockwise.
- `int cucul_stretch_right (cucul_canvas_t *)`
Rotate and stretch a canvas, 90 degrees clockwise.

2.4.1 Detailed Description

These functions perform horizontal and vertical canvas flipping.

2.4.2 Function Documentation

2.4.2.1 `int cucul_invert (cucul_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:

`cv` The canvas to invert.

Returns:

This function always returns 0.

2.4.2.2 `int cucul_flip (cucul_canvas_t * cv)`

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

This function never fails.

Parameters:

`cv` The canvas to flip.

Returns:

This function always returns 0.

References CUCUL_MAGIC_FULLWIDTH.

2.4.2.3 int cucul_flop (cucul_canvas_t * cv)

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

This function never fails.

Parameters:

cv The canvas to flop.

Returns:

This function always returns 0.

2.4.2.4 int cucul_rotate_180 (cucul_canvas_t * cv)

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

This function never fails.

Parameters:

cv The canvas to rotate.

Returns:

This function always returns 0.

References CUCUL_MAGIC_FULLWIDTH.

2.4.2.5 int cucul_rotate_left (cucul_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.

Parameters:

cv The canvas to rotate left.

Returns:

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

2.4.2.6 int cucul_rotate_right (cucul_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.

Parameters:

cv The canvas to rotate right.

Returns:

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

2.4.2.7 int cucul_stretch_left (cucul_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.

Parameters:

cv The canvas to rotate left.

Returns:

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

2.4.2.8 int cucul_stretch_right (cucul_canvas_t * cv)

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

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

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

- **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.

Parameters:

`cv` The canvas to rotate right.

Returns:

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

2.5 libcucul attribute conversions

Functions

- unsigned char `cucul_attr_to_ansi` (unsigned long int)
Get DOS ANSI information from attribute.
- unsigned char `cucul_attr_to_ansi_fg` (unsigned long int)
Get ANSI foreground information from attribute.
- unsigned char `cucul_attr_to_ansi_bg` (unsigned long int)
Get ANSI background information from attribute.
- unsigned int `cucul_attr_to_rgb12_fg` (unsigned long int)
Get 12-bit RGB foreground information from attribute.
- unsigned int `cucul_attr_to_rgb12_bg` (unsigned long int)
Get 12-bit RGB background information from attribute.
- void `cucul_attr_to_argb64` (unsigned long int, unsigned char[8])
Get 64-bit ARGB information from attribute.

2.5.1 Detailed Description

These functions perform conversions between attribute values.

2.5.2 Function Documentation

2.5.2.1 unsigned char `cucul_attr_to_ansi` (unsigned long int *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 `CUCUL_DEFAULT` and `CUCUL_TRANSPARENT` are not handled and are both replaced with `CUCUL_LIGHTGRAY` for the foreground colour and `CUCUL_BLACK` for the background colour.

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

Parameters:

attr The requested attribute value.

Returns:

The corresponding DOS ANSI value.

References CUCUL_BLACK, and CUCUL_LIGHTGRAY.

2.5.2.2 unsigned char cucul_attr_to_ansi_fg (unsigned long int attr)

Get the ANSI foreground colour value for a given attribute. The returned value is either one of the *CUCUL_RED*, *CUCUL_BLACK* etc. predefined colours, or the special value *CUCUL_DEFAULT* meaning the media's default foreground value, or the special value *CUCUL_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:

attr The requested attribute value.

Returns:

The corresponding ANSI foreground value.

2.5.2.3 unsigned char cucul_attr_to_ansi_bg (unsigned long int attr)

Get the ANSI background colour value for a given attribute. The returned value is either one of the *CUCUL_RED*, *CUCUL_BLACK* etc. predefined colours, or the special value *CUCUL_DEFAULT* meaning the media's default background value, or the special value *CUCUL_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:

attr The requested attribute value.

Returns:

The corresponding ANSI background value.

2.5.2.4 unsigned int cucul_attr_to_rgb12_fg (unsigned long int 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:

attr The requested attribute value.

Returns:

The corresponding 12-bit RGB foreground value.

References CUCUL_DEFAULT, CUCUL_LIGHTGRAY, and CUCUL_TRANSPARENT.

2.5.2.5 unsigned int cucul_attr_to_rgb12_bg (unsigned long int 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:

attr The requested attribute value.

Returns:

The corresponding 12-bit RGB background value.

References CUCUL_BLACK, CUCUL_DEFAULT, and CUCUL_TRANSPARENT.

2.5.2.6 void cucul_attr_to_argb64 (unsigned long int attr, unsigned char 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:

- attr* The requested attribute value.
argb An array of 8-bit integers.

References CUCUL_BLACK, CUCUL_DEFAULT, CUCUL_LIGHTGRAY, and CUCUL_TRANSPARENT.

Referenced by `cucul_render_canvas()`.

2.6 libcucul character set conversions

Functions

- unsigned long int `cucul_utf8_to_utf32` (char const *, unsigned int *)
Convert a UTF-8 character to UTF-32.
- unsigned int `cucul_utf32_to_utf8` (char *, unsigned long int)
Convert a UTF-32 character to UTF-8.
- unsigned char `cucul_utf32_to_cp437` (unsigned long int)
Convert a UTF-32 character to CP437.
- unsigned long int `cucul_cp437_to_utf32` (unsigned char)
Convert a CP437 character to UTF-32.
- char `cucul_utf32_to_ascii` (unsigned long int)
Convert a UTF-32 character to ASCII.
- int `cucul_utf32_is_fullwidth` (unsigned long int)
Tell whether a UTF-32 character is fullwidth.

2.6.1 Detailed Description

These functions perform conversions between usual character sets.

2.6.2 Function Documentation

2.6.2.1 unsigned long int `cucul_utf8_to_utf32` (char const * *s*, unsigned int * *read*)

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:

s A string containing the UTF-8 character.

read A pointer to an unsigned integer 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 `cucul_put_str()`.

2.6.2.2 unsigned int cucul_utf32_to_utf8 (char * *buf*, unsigned long int *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:

buf A pointer to a character buffer where the UTF-8 sequence will be written.

ch The UTF-32 character.

Returns:

The number of bytes written.

2.6.2.3 unsigned char cucul_utf32_to_cp437 (unsigned long int *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:

ch The UTF-32 character.

Returns:

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

2.6.2.4 unsigned long int cucul_cp437_to_utf32 (unsigned char *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:

ch The CP437 character.

Returns:

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

2.6.2.5 char cucul_utf32_to_ascii (unsigned long int *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:

ch The UTF-32 character.

Returns:

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

2.6.2.6 int cucul_utf32_is_fullwidth (unsigned long int *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:

ch The UTF-32 character.

Returns:

1 if the character is fullwidth, 0 otherwise.

Referenced by cucul_put_char(), and cucul_put_str().

2.7 libcucul primitives drawing

Functions

- int [cucul_draw_line](#) (cucul_canvas_t *, int, int, int, int, unsigned long int)
Draw a line on the canvas using the given character.
- int [cucul_draw_polyline](#) (cucul_canvas_t *, int const x[], int const y[], int, unsigned long int)
Draw a polyline.
- int [cucul_draw_thin_line](#) (cucul_canvas_t *, int, int, int, int)
Draw a thin line on the canvas, using ASCII art.
- int [cucul_draw_thin_polyline](#) (cucul_canvas_t *, int const x[], int const y[], int)
Draw an ASCII art thin polyline.
- int [cucul_draw_circle](#) (cucul_canvas_t *, int, int, int, unsigned long int)
Draw a circle on the canvas using the given character.
- int [cucul_draw_ellipse](#) (cucul_canvas_t *, int, int, int, int, unsigned long int)

Draw an ellipse on the canvas using the given character.

- int `cucul_draw_thin_ellipse` (`cucul_canvas_t *`, int, int, int, int)

Draw a thin ellipse on the canvas.

- int `cucul_fill_ellipse` (`cucul_canvas_t *`, int, int, int, int, unsigned long int)

Fill an ellipse on the canvas using the given character.

- int `cucul_draw_box` (`cucul_canvas_t *`, int, int, int, int, unsigned long int)

Draw a box on the canvas using the given character.

- int `cucul_draw_thin_box` (`cucul_canvas_t *`, int, int, int, int)

Draw a thin box on the canvas.

- int `cucul_draw_cp437_box` (`cucul_canvas_t *`, int, int, int, int)

Draw a box on the canvas using CP437 characters.

- int `cucul_fill_box` (`cucul_canvas_t *`, int, int, int, int, unsigned long int)

Fill a box on the canvas using the given character.

- int `cucul_draw_triangle` (`cucul_canvas_t *`, int, int, int, int, int, int, unsigned long int)

Draw a triangle on the canvas using the given character.

- int `cucul_draw_thin_triangle` (`cucul_canvas_t *`, int, int, int, int, int, int)

Draw a thin triangle on the canvas.

- int `cucul_fill_triangle` (`cucul_canvas_t *`, int, int, int, int, int, int, unsigned long int)

Fill a triangle on the canvas using the given character.

2.7.1 Detailed Description

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

2.7.2 Function Documentation

2.7.2.1 int `cucul_draw_line` (`cucul_canvas_t *` *cv*, int *x1*, int *y1*, int *x2*, int *y2*, unsigned long int *ch*)

This function never fails.

Parameters:

- cv* The handle to the libcucul canvas.
- x1* X coordinate of the first point.
- y1* Y coordinate of the first point.
- x2* X coordinate of the second point.
- y2* Y coordinate of the second point.
- ch* UTF-32 character to be used to draw the line.

Returns:

This function always returns 0.

Referenced by `cucul_draw_box()`, `cucul_draw_triangle()`, and `cucul_fill_ellipse()`.

2.7.2.2 int cucul_draw_polyline (cucul_canvas_t * cv, int const x[], int const y[], int n, unsigned long int 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:

- cv* The handle to the libcucul canvas.
- x* Array of X coordinates. Must have $n + 1$ elements.
- y* Array of Y coordinates. Must have $n + 1$ elements.
- n* Number of lines to draw.
- ch* UTF-32 character to be used to draw the lines.

Returns:

This function always returns 0.

2.7.2.3 int cucul_draw_thin_line (cucul_canvas_t * cv, int x1, int y1, int x2, int y2)

This function never fails.

Parameters:

- cv* The handle to the libcucul canvas.
- x1* X coordinate of the first point.
- y1* Y coordinate of the first point.
- x2* X coordinate of the second point.
- y2* Y coordinate of the second point.

Returns:

This function always returns 0.

Referenced by `cucul_draw_thin_triangle()`.

2.7.2.4 int cucul_draw_thin_polyline (cucul_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.

Parameters:

- cv* The handle to the libcucul canvas.
- x* Array of X coordinates. Must have $n + 1$ elements.
- y* Array of Y coordinates. Must have $n + 1$ elements.
- n* Number of lines to draw.

Returns:

This function always returns 0.

2.7.2.5 int cucul_draw_circle (cucul_canvas_t * *cv*, int *x*, int *y*, int *r*, unsigned long int *ch*)

This function never fails.

Parameters:

- cv* The handle to the libcucul canvas.
- x* Center X coordinate.
- y* Center Y coordinate.
- r* Circle radius.
- ch* UTF-32 character to be used to draw the circle outline.

Returns:

This function always returns 0.

2.7.2.6 int cucul_draw_ellipse (cucul_canvas_t * *cv*, int *xo*, int *yo*, int *a*, int *b*, unsigned long int *ch*)

This function never fails.

Parameters:

- cv* The handle to the libcucul canvas.
- xo* Center X coordinate.
- yo* Center Y coordinate.
- a* Ellipse X radius.
- b* Ellipse Y radius.
- ch* UTF-32 character to be used to draw the ellipse outline.

Returns:

This function always returns 0.

2.7.2.7 int cucul_draw_thin_ellipse (cucul_canvas_t * *cv*, int *xo*, int *yo*, int *a*, int *b*)

This function never fails.

Parameters:

- cv* The handle to the libcucul canvas.
- xo* Center X coordinate.
- yo* Center Y coordinate.
- a* Ellipse X radius.
- b* Ellipse Y radius.

Returns:

This function always returns 0.

2.7.2.8 int cucul_fill_ellipse (cucul_canvas_t * *cv*, int *xo*, int *yo*, int *a*, int *b*, unsigned long int *ch*)

This function never fails.

Parameters:

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

Returns:

This function always returns 0.

References cucul_draw_line().

2.7.2.9 int cucul_draw_box (cucul_canvas_t * *cv*, int *xl*, int *yl*, int *w*, int *h*, unsigned long int *ch*)

This function never fails.

Parameters:

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

Returns:

This function always returns 0.

References cucul_draw_line().

2.7.2.10 int cucul_draw_thin_box (cucul_canvas_t * *cv*, int *x1*, int *y1*, int *w*, int *h*)

This function never fails.

Parameters:

- cv* The handle to the libcucul canvas.
- x* X coordinate of the upper-left corner of the box.
- y* Y coordinate of the upper-left corner of the box.
- w* Width of the box.
- h* Height of the box.

Returns:

This function always returns 0.

References cucul_put_char().

2.7.2.11 int cucul_draw_cp437_box (cucul_canvas_t * *cv*, int *x1*, int *y1*, int *w*, int *h*)

This function never fails.

Parameters:

- cv* The handle to the libcucul canvas.
- x* X coordinate of the upper-left corner of the box.
- y* Y coordinate of the upper-left corner of the box.
- w* Width of the box.
- h* Height of the box.

Returns:

This function always returns 0.

References cucul_put_char().

2.7.2.12 int cucul_fill_box (cucul_canvas_t * *cv*, int *x1*, int *y1*, int *w*, int *h*, unsigned long int *ch*)

This function never fails.

Parameters:

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

Returns:

This function always returns 0.

References cucul_put_char().

2.7.2.13 int cucul_draw_triangle (cucul_canvas_t * *cv*, int *x1*, int *y1*, int *x2*, int *y2*, int *x3*, int *y3*, unsigned long int *ch*)

This function never fails.

Parameters:

- cv* The handle to the libcucul canvas.
- x1* X coordinate of the first point.
- y1* Y coordinate of the first point.
- x2* X coordinate of the second point.
- y2* Y coordinate of the second point.
- x3* X coordinate of the third point.
- y3* Y coordinate of the third point.
- ch* UTF-32 character to be used to draw the triangle outline.

Returns:

This function always returns 0.

References cucul_draw_line().

2.7.2.14 int cucul_draw_thin_triangle (cucul_canvas_t * *cv*, int *x1*, int *y1*, int *x2*, int *y2*, int *x3*, int *y3*)

This function never fails.

Parameters:

- cv* The handle to the libcucul canvas.
- x1* X coordinate of the first point.
- y1* Y coordinate of the first point.
- x2* X coordinate of the second point.
- y2* Y coordinate of the second point.
- x3* X coordinate of the third point.
- y3* Y coordinate of the third point.

Returns:

This function always returns 0.

References cucul_draw_thin_line().

2.7.2.15 int cucul_fill_triangle (cucul_canvas_t * *cv*, int *x1*, int *y1*, int *x2*, int *y2*, int *x3*, int *y3*, unsigned long int *ch*)

This function never fails.

Parameters:

- cv* The handle to the libcucul canvas.
- x1* X coordinate of the first point.

y1 Y coordinate of the first point.
x2 X coordinate of the second point.
y2 Y coordinate of the second point.
x3 X coordinate of the third point.
y3 Y coordinate of the third point.
ch UTF-32 character to be used to fill the triangle.

Returns:

This function always returns 0.

References `cucul_fill_triangle()`, and `cucul_put_char()`.

Referenced by `cucul_fill_triangle()`.

2.8 libcucul canvas frame handling

Functions

- unsigned int `cucul_get_frame_count` (`cucul_canvas_t` const *)
Get the number of frames in a canvas.
- int `cucul_set_frame` (`cucul_canvas_t` *, unsigned int)
Activate a given canvas frame.
- char const * `cucul_get_frame_name` (`cucul_canvas_t` const *)
Get the current frame's name.
- int `cucul_set_frame_name` (`cucul_canvas_t` *, char const *)
Set the current frame's name.
- int `cucul_create_frame` (`cucul_canvas_t` *, unsigned int)
Add a frame to a canvas.
- int `cucul_free_frame` (`cucul_canvas_t` *, unsigned int)
Remove a frame from a canvas.

2.8.1 Detailed Description

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

2.8.2 Function Documentation

2.8.2.1 unsigned int `cucul_get_frame_count` (`cucul_canvas_t` const * *cv*)

Return the current canvas' frame count.

This function never fails.

Parameters:

cv A libcucul canvas

Returns:

The frame count

Referenced by `cucul_set_canvas_boundaries()`.

2.8.2.2 int cucul_set_frame (cucul_canvas_t * *cv*, unsigned int *id*)

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

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

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

- `EINVAL` Requested frame is out of range.

Parameters:

cv A libcucul canvas

id The canvas frame to activate

Returns:

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

Referenced by `cucul_set_canvas_boundaries()`.

2.8.2.3 char const* cucul_get_frame_name (cucul_canvas_t const * *cv*)

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

This function never fails.

Parameters:

cv A libcucul canvas.

Returns:

The current frame's name.

2.8.2.4 int cucul_set_frame_name (cucul_canvas_t * *cv*, char const * *name*)

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

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

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

Parameters:

cv A libcucul canvas.

name The name to give to the current frame.

Returns:

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

2.8.2.5 int cucul_create_frame (cucul_canvas_t * cv, unsigned 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.

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:

cv A libcucul canvas

id The index where to insert the new frame

Returns:

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

Referenced by cucul_set_canvas_boundaries().

2.8.2.6 int cucul_free_frame (cucul_canvas_t * cv, unsigned 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.

Parameters:

cv A libcucul canvas

id The index of the frame to delete

Returns:

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

2.9 libcucul bitmap dithering

Functions

- `cucul_dither_t * cucul_create_dither` (unsigned int, unsigned int, unsigned int, unsigned int, unsigned long int, unsigned long int, unsigned long int, unsigned long int)
Create an internal dither object.
- `int cucul_set_dither_palette` (`cucul_dither_t *`, unsigned int `r[]`, unsigned int `g[]`, unsigned int `b[]`, unsigned int `a[]`)
Set the palette of an 8bpp dither object.
- `int cucul_set_dither_brightness` (`cucul_dither_t *`, float)
Set the brightness of a dither object.
- `float cucul_get_dither_brightness` (`cucul_dither_t` const *)
Get the brightness of a dither object.
- `int cucul_set_dither_gamma` (`cucul_dither_t *`, float)
Set the gamma of a dither object.
- `float cucul_get_dither_gamma` (`cucul_dither_t` const *)
Get the gamma of a dither object.
- `int cucul_set_dither_contrast` (`cucul_dither_t *`, float)
Set the contrast of a dither object.
- `float cucul_get_dither_contrast` (`cucul_dither_t` const *)
Get the contrast of a dither object.
- `int cucul_set_dither_antialias` (`cucul_dither_t *`, char const *)
Set dither antialiasing.
- `char const *const * cucul_get_dither_antialias_list` (`cucul_dither_t` const *)
Get available antialiasing methods.
- `char const * cucul_get_dither_antialias` (`cucul_dither_t` const *)
Get current antialiasing method.
- `int cucul_set_dither_color` (`cucul_dither_t *`, char const *)
Choose colours used for dithering.
- `char const *const * cucul_get_dither_color_list` (`cucul_dither_t` const *)
Get available colour modes.
- `char const * cucul_get_dither_color` (`cucul_dither_t` const *)
Get current colour mode.
- `int cucul_set_dither_charset` (`cucul_dither_t *`, char const *)
Choose characters used for dithering.

- `char const *const * cucul_get_dither_charset_list (cucul_dither_t const *)`
Get available dither character sets.
- `char const * cucul_get_dither_charset (cucul_dither_t const *)`
Get current character set.
- `int cucul_set_dither_algorithm (cucul_dither_t *, char const *)`
Set dithering algorithm.
- `char const *const * cucul_get_dither_algorithm_list (cucul_dither_t const *)`
Get dithering algorithms.
- `char const * cucul_get_dither_algorithm (cucul_dither_t const *)`
Get current dithering algorithm.
- `int cucul_dither_bitmap (cucul_canvas_t *, int, int, int, int, cucul_dither_t const *, void *)`
Dither a bitmap on the canvas.
- `int cucul_free_dither (cucul_dither_t *)`
Free the memory associated with a dither.

2.9.1 Detailed Description

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

2.9.2 Function Documentation

2.9.2.1 `cucul_dither_t* cucul_create_dither (unsigned int bpp, unsigned int w, unsigned int h, unsigned int pitch, unsigned long int rmask, unsigned long int gmask, unsigned long int bmask, unsigned long int 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 `cucul_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.

Parameters:

- bpp*** Bitmap depth in bits per pixel.
- w*** Bitmap width in pixels.
- h*** Bitmap height in pixels.
- pitch*** Bitmap pitch in bytes.
- rmask*** Bitmask for red values.

gmask Bitmask for green values.

bmask Bitmask for blue values.

amask Bitmask for alpha values.

Returns:

Dither object upon success, NULL if an error occurred.

2.9.2.2 int cucul_set_dither_palette (cucul_dither_t * *d*, unsigned int *red*[], unsigned int *green*[], unsigned int *blue*[], unsigned int *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.

Parameters:

d Dither object.

red Array of 256 red values.

green Array of 256 green values.

blue Array of 256 blue values.

alpha Array of 256 alpha values.

Returns:

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

2.9.2.3 int cucul_set_dither_brightness (cucul_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.

Parameters:

d Dither object.

brightness brightness value.

Returns:

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

2.9.2.4 float cucul_get_dither_brightness (cucul_dither_t const * *d*)

Get the brightness of the given dither object.

This function never fails.

Parameters:

d Dither object.

Returns:

Brightness value.

2.9.2.5 int cucul_set_dither_gamma (cucul_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.

Parameters:

d Dither object.

gamma Gamma value.

Returns:

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

2.9.2.6 float cucul_get_dither_gamma (cucul_dither_t const * *d*)

Get the gamma of the given dither object.

This function never fails.

Parameters:

d Dither object.

Returns:

Gamma value.

2.9.2.7 int cucul_set_dither_contrast (cucul_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.

Parameters:

d Dither object.

contrast contrast value.

Returns:

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

2.9.2.8 float cucul_get_dither_contrast (cucul_dither_t const * *d*)

Get the contrast of the given dither object.

This function never fails.

Parameters:

d Dither object.

Returns:

Contrast value.

2.9.2.9 int cucul_set_dither_antialias (cucul_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.

Parameters:

d Dither object.

str A string describing the antialiasing method that will be used for the dithering.

Returns:

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

2.9.2.10 char const* const* cucul_get_dither_antialias_list (cucul_dither_t const * *d*)

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

This function never fails.

Parameters:

d Dither object.

Returns:

An array of strings.

2.9.2.11 char const* cucul_get_dither_antialias (cucul_dither_t const * *d*)

Return the given dither's current antialiasing method.

This function never fails.

Parameters:

d Dither object.

Returns:

A static string.

2.9.2.12 int cucul_set_dither_color (cucul_dither_t * *d*, char const * *str*)

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

- "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.

Parameters:

d Dither object.

str A string describing the colour set that will be used for the dithering.

Returns:

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

2.9.2.13 char const* const* cucul_get_dither_color_list (cucul_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 [cucul_set_dither_color\(\)](#), and a string containing the natural language description for that colour mode.

This function never fails.

Parameters:

d Dither object.

Returns:

An array of strings.

2.9.2.14 char const* cucul_get_dither_color (cucul_dither_t const * *d*)

Return the given dither's current colour mode.

This function never fails.

Parameters:

d Dither object.

Returns:

A static string.

2.9.2.15 int cucul_set_dither_charset (cucul_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.

Parameters:

d Dither object.

str A string describing the characters that need to be used for the dithering.

Returns:

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

2.9.2.16 char const* const* cucul_get_dither_charset_list (cucul_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, to be used with [cucul_set_dither_charset\(\)](#), and a string containing the natural language description for that character set.

This function never fails.

Parameters:

d Dither object.

Returns:

An array of strings.

2.9.2.17 char const* cucul_get_dither_charset (cucul_dither_t const * *d*)

Return the given dither's current character set.

This function never fails.

Parameters:

d Dither object.

Returns:

A static string.

2.9.2.18 int cucul_set_dither_algorithm (cucul_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.

Parameters:

d Dither object.

str A string describing the algorithm that needs to be used for the dithering.

Returns:

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

2.9.2.19 char const* const* cucul_get_dither_algorithm_list (cucul_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 `cucul_set_dither_dithering()`, and a string containing the natural language description for that algorithm.

This function never fails.

Parameters:

d Dither object.

Returns:

An array of strings.

2.9.2.20 char const* cucul_get_dither_algorithm (cucul_dither_t const * *d*)

Return the given dither's current dithering algorithm.

This function never fails.

Parameters:

d Dither object.

Returns:

A static string.

2.9.2.21 int cucul_dither_bitmap (cucul_canvas_t * *cv*, int *x*, int *y*, int *w*, int *h*, cucul_dither_t const * *d*, void * *pixels*)

Dither a bitmap at the given coordinates. The dither can be of any size and will be stretched to the text area.

This function never fails.

Parameters:

cv A handle to the libcucul canvas.

x X coordinate of the upper-left corner of the drawing area.

y Y coordinate of the upper-left corner of the drawing area.

w Width of the drawing area.

h Height of the drawing area.

d Dither object to be drawn.

pixels Bitmap's pixels.

Returns:

This function always returns 0.

References CUCUL_BLACK, cucul_get_attr(), cucul_put_char(), cucul_set_attr(), and cucul_set_color_ansi().

2.9.2.22 int cucul_free_dither (cucul_dither_t * *d*)

Free the memory allocated by [cucul_create_dither\(\)](#).

This function never fails.

Parameters:

d Dither object.

Returns:

This function always returns 0.

2.10 libcucul font handling

Functions

- `cucul_font_t * cucul_load_font` (void const *, unsigned int)
Load a font from memory for future use.
- `char const *const * cucul_get_font_list` (void)
Get available builtin fonts.
- `unsigned int cucul_get_font_width` (cucul_font_t const *)
Get a font's standard glyph width.
- `unsigned int cucul_get_font_height` (cucul_font_t const *)
Get a font's standard glyph height.
- `unsigned long int const * cucul_get_font_blocks` (cucul_font_t const *)
Get a font's list of supported glyphs.
- `int cucul_render_canvas` (cucul_canvas_t const *, cucul_font_t const *, void *, unsigned int, unsigned int, unsigned int)
Render the canvas onto an image buffer.
- `int cucul_free_font` (cucul_font_t *)
Free a font structure.

2.10.1 Detailed Description

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

2.10.2 Function Documentation

2.10.2.1 `cucul_font_t* cucul_load_font` (void const * *data*, unsigned int *size*)

This function loads a font and returns a handle to its internal structure. The handle can then be used with `cucul_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 `cucul_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:

data The memory area containing the font or its name.

size 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 `cucul_load_font()`.

Referenced by `cucul_load_font()`.

2.10.2.2 char const* const* cucul_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.

2.10.2.3 unsigned int cucul_get_font_width (cucul_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:

f The font, as returned by `cucul_load_font()`

Returns:

The standard glyph width.

2.10.2.4 unsigned int cucul_get_font_height (cucul_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:

f The font, as returned by `cucul_load_font()`

Returns:

The standard glyph height.

2.10.2.5 unsigned long int const* cucul_get_font_blocks (cucul_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:

f The font, as returned by [cucul_load_font\(\)](#)

Returns:

The list of Unicode blocks supported by the font.

2.10.2.6 int cucul_render_canvas (cucul_canvas_t const * *cv*, cucul_font_t const * *f*, void * *buf*, unsigned int *width*, unsigned int *height*, unsigned 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 [cucul_get_canvas_width\(\)](#) and [cucul_get_font_width\(\)](#). The required height can be computed using [cucul_get_canvas_height\(\)](#) and [cucul_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.

This function never fails.

Parameters:

cv The canvas to render

f The font, as returned by [cucul_load_font\(\)](#)

buf The image buffer

width The width (in pixels) of the image buffer

height The height (in pixels) of the image buffer

pitch The pitch (in bytes) of an image buffer line.

Returns:

This function always returns 0.

References [cucul_attr_to_argb64\(\)](#).

2.10.2.7 int cucul_free_font (cucul_font_t * *f*)

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

This function never fails.

Parameters:

f The font, as returned by [cucul_load_font\(\)](#)

Returns:

This function always returns 0.

2.11 libcucul importers/exporters from/to various**Functions**

- long int `cucul_import_memory` (`cucul_canvas_t *`, void const *, unsigned long int, char const *)
Import a memory buffer into a canvas.
- long int `cucul_import_file` (`cucul_canvas_t *`, char const *, char const *)
Import a file into a canvas.
- char const *const * `cucul_get_import_list` (void)
Get available import formats.
- void * `cucul_export_memory` (`cucul_canvas_t` const *, char const *, unsigned long int *)
Export a canvas into a foreign format.
- char const *const * `cucul_get_export_list` (void)
Get available export formats.

2.11.1 Detailed Description

formats

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

2.11.2 Function Documentation**2.11.2.1 long int `cucul_import_memory` (`cucul_canvas_t *` *cv*, void const * *data*, unsigned long int *len*, char const * *format*)**

Import a memory buffer into the given libcucul 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.
- `"ansi"`: import ANSI files.
- `"utf8"`: import UTF-8 files with ANSI colour codes.

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.

Parameters:

cv A libcucul canvas in which to import the file.
data A memory area containing the data to be loaded into the canvas.
len The size in bytes of the memory area.
format 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.

Referenced by `cucul_import_file()`.

2.11.2.2 long int cucul_import_file (cucul_canvas_t *cv, char const *filename, char const *format)

Import a file into the given libcucul 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.
- `"ansi"`: import ANSI files.
- `"utf8"`: import UTF-8 files with ANSI colour codes.

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. `cucul_import_file()` may also fail and set **errno** for any of the errors specified for the routine `fopen()`.

Parameters:

cv A libcucul canvas in which to import the file.
filename The name of the file to load.
format 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 `cucul_import_memory()`.

2.11.2.3 char const* const* cucul_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 `cucul_import_canvas()`, and a string containing the natural language description for that import format.

This function never fails.

Returns:

An array of strings.

2.11.2.4 void* cucul_export_memory (cucul_canvas_t const * *cv*, char const * *format*, unsigned long int * *bytes*)

This function exports a libcucul 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.

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

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

Parameters:

cv A libcucul canvas

format A string describing the requested output format.

bytes A pointer to an unsigned long integer where the number of allocated bytes will be written.

Returns:

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

2.11.2.5 char const* const* cucul_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 [cucul_export_memory\(\)](#), and a string containing the natural language description for that export format.

This function never fails.

Returns:

An array of strings.

2.12 libcaca basic functions

Functions

- [caca_display_t](#) * [caca_create_display](#) ([cucul_canvas_t](#) *)
Attach a caca graphical context to a cucul canvas.
- int [caca_free_display](#) ([caca_display_t](#) *)
Detach a caca graphical context from a cucul backend context.
- [cucul_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](#) *, unsigned int)
Set the refresh delay.
- unsigned int [caca_get_display_time](#) ([caca_display_t](#) const *)
Get the display's average rendering time.
- unsigned int [caca_get_display_width](#) ([caca_display_t](#) const *)
Get the display width.
- unsigned 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.
- char const * [caca_get_version](#) (void)
Return the libcaca version.

2.12.1 Detailed Description

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

2.12.2 Function Documentation

2.12.2.1 `caca_display_t* caca_create_display (cucul_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 libcucul canvas. Everything that gets drawn in the libcucul canvas can then be displayed by the libcaca driver.

If no cucul 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 an error occurs, NULL is returned and `errno` is set accordingly:

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

Parameters:

cv The cucul canvas.

Returns:

The caca graphical context or NULL if an error occurred.

References CACA_EVENT_NONE, cucul_create_canvas(), cucul_free_canvas(), cucul_get_canvas_height(), cucul_get_canvas_width(), cucul_manage_canvas(), and cucul_unmanage_canvas().

2.12.2.2 `int caca_free_display (caca_display_t * dp)`

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

If the cucul canvas was automatically created by `caca_create_display()`, it is automatically destroyed and any handle to it becomes invalid.

This function never fails.

Parameters:

dp The libcaca graphical context.

Returns:

This function always returns 0.

References cucul_free_canvas(), and cucul_unmanage_canvas().

2.12.2.3 `cucul_canvas_t* caca_get_canvas (caca_display_t * dp)`

Return a handle on the *cucul_canvas_t* object that was either attached or created by `caca_create_display()`.

This function never fails.

Parameters:

dp The libcaca graphical context.

Returns:

The libcucul canvas.

2.12.2.4 int caca_refresh_display (caca_display_t * *dp*)

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

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

This function never fails.

Parameters:

dp The libcaca display context.

Returns:

This function always returns 0.

2.12.2.5 int caca_set_display_time (caca_display_t * *dp*, unsigned int *usec*)

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

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

This function never fails.

Parameters:

dp The libcaca display context.

usec The refresh delay in microseconds.

Returns:

This function always returns 0.

2.12.2.6 unsigned 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.

Parameters:

dp The libcaca display context.

Returns:

The render time in microseconds.

2.12.2.7 unsigned int caca_get_display_width (caca_display_t const * *dp*)

If libcacaca runs in a window, get the usable window width. This value can be used for aspect ratio calculation. If libcacaca 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.

Parameters:

dp The libcacaca display context.

Returns:

The display width.

2.12.2.8 unsigned int caca_get_display_height (caca_display_t const * *dp*)

If libcacaca runs in a window, get the usable window height. This value can be used for aspect ratio calculation. If libcacaca 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.

Parameters:

dp The libcacaca display context.

Returns:

The display height.

2.12.2.9 int caca_set_display_title (caca_display_t * *dp*, char const * *title*)

If libcacaca 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:

dp The libcacaca display context.

title The desired display title.

Returns:

0 upon success, -1 if an error occurred.

2.12.2.10 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:

- dp* The libcaca display context.
- flag* 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.

2.12.2.11 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.

Parameters:

- dp* The libcaca display context.
- flag* 0 hides the cursor, 1 shows the system's default cursor (usually a white rectangle). Other values are reserved for future use.

Returns:

0 upon success, -1 if an error occurred.

2.12.2.12 char const* caca_get_version (void)

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

This function never fails.

Returns:

The *libcaca* version information.

2.13 libcaca event handling**Functions**

- int `caca_get_event` (`caca_display_t *`, unsigned int, `caca_event_t *`, int)
Get the next mouse or keyboard input event.
- unsigned int `caca_get_mouse_x` (`caca_display_t` const *)
Return the X mouse coordinate.
- unsigned 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.
- unsigned int `caca_get_event_key_ch` (`caca_event_t` const *)
Return a key press or key release event's value.
- unsigned long int `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.
- unsigned int `caca_get_event_mouse_button` (`caca_event_t` const *)
Return a mouse press or mouse release event's button.
- unsigned int `caca_get_event_mouse_x` (`caca_event_t` const *)
Return a mouse motion event's X coordinate.
- unsigned int `caca_get_event_mouse_y` (`caca_event_t` const *)
Return a mouse motion event's Y coordinate.
- unsigned int `caca_get_event_resize_width` (`caca_event_t` const *)
Return a resize event's display width value.
- unsigned int `caca_get_event_resize_height` (`caca_event_t` const *)
Return a resize event's display height value.

2.13.1 Detailed Description

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

2.13.2 Function Documentation

2.13.2.1 int `caca_get_event` (`caca_display_t` * *dp*, unsigned 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.

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.

Parameters:

dp The libcaca graphical context.

event_mask Bitmask of requested events.

timeout A timeout value in microseconds, -1 for blocking behaviour

ev A pointer to a [caca_event](#) structure, or NULL.

Returns:

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

References CACA_EVENT_NONE.

2.13.2.2 unsigned 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.

Parameters:

dp The libcaca graphical context.

Returns:

The X mouse coordinate.

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

2.13.2.3 unsigned int caca_get_mouse_y (caca_display_t const * dp)

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

This function never fails.

Parameters:

dp The libcaca graphical context.

Returns:

The Y mouse coordinate.

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

2.13.2.4 enum caca_event_type caca_get_event_type (caca_event_t const * ev)

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

- CACA_EVENT_NONE: no other function may be called.
- CACA_EVENT_KEY_PRESS, CACA_EVENT_KEY_RELEASE: [caca_get_event_key_ch\(\)](#), [caca_get_event_key_utf32\(\)](#) and [caca_get_event_key_utf8\(\)](#) may be called.
- CACA_EVENT_MOUSE_PRESS, CACA_EVENT_MOUSE_RELEASE: [caca_get_event_mouse_button\(\)](#) may be called.

- `CACA_EVENT_MOUSE_MOTION`: `caca_get_event_mouse_x()` and `caca_get_event_mouse_y()` may be called.
- `CACA_EVENT_RESIZE`: `caca_get_event_resize_width()` and `caca_get_event_resize_height()` may be called.
- `CACA_EVENT_QUIT`: no other function may be called.

This function never fails.

Parameters:

ev The libcaca event.

Returns:

The event's type.

2.13.2.5 `unsigned int caca_get_event_key_ch (caca_event_t const * ev)`

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

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.

Parameters:

ev The libcaca event.

Returns:

The key value.

2.13.2.6 `unsigned long int 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.

Parameters:

ev The libcaca event.

Returns:

The key's Unicode value.

2.13.2.7 `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.

Parameters:

ev The libcaca event.

Returns:

This function always returns 0.

2.13.2.8 unsigned 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.

Parameters:

ev The libcaca event.

Returns:

The event's mouse button.

2.13.2.9 unsigned 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.

Parameters:

ev The libcaca event.

Returns:

The event's X mouse coordinate.

2.13.2.10 unsigned 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.

Parameters:

ev The libcaca event.

Returns:

The event's Y mouse coordinate.

2.13.2.11 unsigned 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.

Parameters:

ev The libcaca event.

Returns:

The event's new display width value.

2.13.2.12 unsigned 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.

Parameters:

ev The libcaca event.

Returns:

The event's new display height value.

3 Data Structure Documentation

3.1 `caca_event` Struct Reference

Handling of user events.

Data Fields

- unsigned char **opaque_structure** [32]

3.1.1 Detailed Description

This structure is filled by [caca_get_event\(\)](#) when an event is received. It is an opaque structure that should only be accessed through `caca_event_get_type()` and similar functions.

4 File Documentation

4.1 `caca.h` File Reference

The *libcaca* public header.

Data Structures

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

Defines

- #define [__extern](#) extern
- #define [CACAPI_VERSION_1](#)

Typedefs

- typedef struct caca_display [caca_display_t](#)
- typedef struct [caca_event](#) [caca_event_t](#)

Enumerations

- 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.
- enum [caca_key](#) {
[CACAPI_KEY_UNKNOWN](#) = 0x00, [CACAPI_KEY_CTRL_A](#) = 0x01, [CACAPI_KEY_CTRL_B](#) = 0x02, [CACAPI_KEY_CTRL_C](#) = 0x03,
[CACAPI_KEY_CTRL_D](#) = 0x04, [CACAPI_KEY_CTRL_E](#) = 0x05, [CACAPI_KEY_CTRL_F](#) = 0x06, [CACAPI_KEY_CTRL_G](#) = 0x07,
[CACAPI_KEY_BACKSPACE](#) = 0x08, [CACAPI_KEY_TAB](#) = 0x09, [CACAPI_KEY_CTRL_J](#) = 0x0a, [CACAPI_KEY_CTRL_K](#) = 0x0b,
[CACAPI_KEY_CTRL_L](#) = 0x0c, [CACAPI_KEY_RETURN](#) = 0x0d, [CACAPI_KEY_CTRL_N](#) = 0x0e, [CACAPI_KEY_CTRL_O](#) = 0x0f,
[CACAPI_KEY_CTRL_P](#) = 0x10, [CACAPI_KEY_CTRL_Q](#) = 0x11, [CACAPI_KEY_CTRL_R](#) = 0x12, [CACAPI_KEY_PAUSE](#) = 0x13,
[CACAPI_KEY_CTRL_T](#) = 0x14, [CACAPI_KEY_CTRL_U](#) = 0x15, [CACAPI_KEY_CTRL_V](#) = 0x16, [CACAPI_KEY_CTRL_W](#) = 0x17,
[CACAPI_KEY_CTRL_X](#) = 0x18, [CACAPI_KEY_CTRL_Y](#) = 0x19, [CACAPI_KEY_CTRL_Z](#) = 0x1a, [CACAPI_KEY_ESCAPE](#) = 0x1b,
[CACAPI_KEY_DELETE](#) = 0x7f, [CACAPI_KEY_UP](#) = 0x111, [CACAPI_KEY_DOWN](#) = 0x112, [CACAPI_KEY_LEFT](#) = 0x113,
[CACAPI_KEY_RIGHT](#) = 0x114, [CACAPI_KEY_INSERT](#) = 0x115, [CACAPI_KEY_HOME](#) = 0x116, [CACAPI_KEY_END](#) = 0x117,
[CACAPI_KEY_PAGEUP](#) = 0x118, [CACAPI_KEY_PAGEDOWN](#) = 0x119, [CACAPI_KEY_F1](#) = 0x11a, [CACAPI_KEY_F2](#) = 0x11b,


```
CACA_KEY_F3 = 0x11c, CACA_KEY_F4 = 0x11d, CACA_KEY_F5 = 0x11e, CACA_KEY_F6  
= 0x11f,  
CACA_KEY_F7 = 0x120, CACA_KEY_F8 = 0x121, CACA_KEY_F9 = 0x122, CACA_KEY_F10  
= 0x123,  
CACA_KEY_F11 = 0x124, CACA_KEY_F12 = 0x125, CACA_KEY_F13 = 0x126, CACA_KEY_  
F14 = 0x127,  
CACA_KEY_F15 = 0x128 }
```

Special key values.

Functions

- `caca_display_t * caca_create_display (cucul_canvas_t *)`
Attach a caca graphical context to a cucul canvas.
- `int caca_free_display (caca_display_t *)`
Detach a caca graphical context from a cucul backend context.
- `cucul_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 *, unsigned int)`
Set the refresh delay.
- `unsigned int caca_get_display_time (caca_display_t const *)`
Get the display's average rendering time.
- `unsigned int caca_get_display_width (caca_display_t const *)`
Get the display width.
- `unsigned 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.
- `char const * caca_get_version (void)`
Return the libcaca version.
- `int caca_get_event (caca_display_t *, unsigned int, caca_event_t *, int)`

Get the next mouse or keyboard input event.

- unsigned int [caca_get_mouse_x](#) ([caca_display_t](#) const *)
Return the X mouse coordinate.
- unsigned 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.
- unsigned int [caca_get_event_key_ch](#) ([caca_event_t](#) const *)
Return a key press or key release event's value.
- unsigned long int [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.
- unsigned int [caca_get_event_mouse_button](#) ([caca_event_t](#) const *)
Return a mouse press or mouse release event's button.
- unsigned int [caca_get_event_mouse_x](#) ([caca_event_t](#) const *)
Return a mouse motion event's X coordinate.
- unsigned int [caca_get_event_mouse_y](#) ([caca_event_t](#) const *)
Return a mouse motion event's Y coordinate.
- unsigned int [caca_get_event_resize_width](#) ([caca_event_t](#) const *)
Return a resize event's display width value.
- unsigned int [caca_get_event_resize_height](#) ([caca_event_t](#) const *)
Return a resize event's display height value.

4.1.1 Detailed Description

Version:

\$Id\$

Author:

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

4.1.2 Define Documentation

4.1.2.1 #define CACA_API_VERSION_1

libcaca API version

4.1.3 Typedef Documentation

4.1.3.1 typedef struct caca_display caca_display_t

libcaca display context

4.1.3.2 typedef struct caca_event caca_event_t

libcaca event structure

4.1.4 Enumeration Type Documentation

4.1.4.1 enum caca_event_type

This enum serves two purposes:

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

Enumerator:

CACA_EVENT_NONE No event.
CACA_EVENT_KEY_PRESS A key was pressed.
CACA_EVENT_KEY_RELEASE A key was released.
CACA_EVENT_MOUSE_PRESS A mouse button was pressed.
CACA_EVENT_MOUSE_RELEASE A mouse button was released.
CACA_EVENT_MOUSE_MOTION The mouse was moved.
CACA_EVENT_RESIZE The window was resized.
CACA_EVENT_QUIT The user requested to quit.
CACA_EVENT_ANY Bitmask for any event.

4.1.4.2 enum caca_key

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

Enumerator:

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.
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.

4.2 cucul.h File Reference

The *libcucul* public header.

Defines

- #define **__extern** extern
- #define **CUCUL_API_VERSION_1**
- #define **CUCUL_BLACK** 0x00
- #define **CUCUL_BLUE** 0x01
- #define **CUCUL_GREEN** 0x02
- #define **CUCUL_CYAN** 0x03
- #define **CUCUL_RED** 0x04
- #define **CUCUL_MAGENTA** 0x05
- #define **CUCUL_BROWN** 0x06
- #define **CUCUL_LIGHTGRAY** 0x07
- #define **CUCUL_DARKGRAY** 0x08
- #define **CUCUL_LIGHTBLUE** 0x09
- #define **CUCUL_LIGHTGREEN** 0x0a
- #define **CUCUL_LIGHTCYAN** 0x0b
- #define **CUCUL_LIGHTRED** 0x0c
- #define **CUCUL_LIGHTMAGENTA** 0x0d
- #define **CUCUL_YELLOW** 0x0e
- #define **CUCUL_WHITE** 0x0f
- #define **CUCUL_DEFAULT** 0x10
- #define **CUCUL_TRANSPARENT** 0x20
- #define **CUCUL_BOLD** 0x01
- #define **CUCUL_ITALICS** 0x02
- #define **CUCUL_UNDERLINE** 0x04
- #define **CUCUL_BLINK** 0x08
- #define **CUCUL_MAGIC_FULLWIDTH** 0x000ffffe

Typedefs

- typedef struct cucul_canvas **cucul_canvas_t**
- typedef struct cucul_dither **cucul_dither_t**
- typedef struct cucul_font **cucul_font_t**

Functions

- **cucul_canvas_t * cucul_create_canvas** (unsigned int, unsigned int)
Initialise a libcucul canvas.
- **int cucul_manage_canvas** (**cucul_canvas_t ***, int(*) (void *), void *)
Manage a canvas.
- **int cucul_unmanage_canvas** (**cucul_canvas_t ***, int(*) (void *), void *)
Unmanage a canvas.

- int [cucul_set_canvas_size](#) ([cucul_canvas_t](#) *, unsigned int, unsigned int)
Resize a canvas.
- unsigned int [cucul_get_canvas_width](#) ([cucul_canvas_t](#) const *)
Get the canvas width.
- unsigned int [cucul_get_canvas_height](#) ([cucul_canvas_t](#) const *)
Get the canvas height.
- unsigned char const * [cucul_get_canvas_chars](#) ([cucul_canvas_t](#) const *)
Get the canvas character array.
- unsigned char const * [cucul_get_canvas_attrs](#) ([cucul_canvas_t](#) const *)
Get the canvas attribute array.
- int [cucul_free_canvas](#) ([cucul_canvas_t](#) *)
Uninitialise libcucul.
- int [cucul_rand](#) (int, int)
Generate a random integer within a range.
- char const * [cucul_get_version](#) (void)
Return the libcucul version.
- int [cucul_gotoxy](#) ([cucul_canvas_t](#) *, int, int)
Set cursor position.
- int [cucul_get_cursor_x](#) ([cucul_canvas_t](#) const *)
Get X cursor position.
- int [cucul_get_cursor_y](#) ([cucul_canvas_t](#) const *)
Get Y cursor position.
- int [cucul_put_char](#) ([cucul_canvas_t](#) *, int, int, unsigned long int)
Print an ASCII or Unicode character.
- unsigned long int [cucul_get_char](#) ([cucul_canvas_t](#) const *, int, int)
Get the Unicode character at the given coordinates.
- int [cucul_put_str](#) ([cucul_canvas_t](#) *, int, int, char const *)
Print a string.
- unsigned long int [cucul_get_attr](#) ([cucul_canvas_t](#) const *, int, int)
Get the text attribute at the given coordinates.
- int [cucul_set_attr](#) ([cucul_canvas_t](#) *, unsigned long int)
Set the default character attribute.
- int [cucul_put_attr](#) ([cucul_canvas_t](#) *, int, int, unsigned long int)
Set the character attribute at the given coordinates.

- int `cucul_set_color_ansi` (`cucul_canvas_t *`, unsigned char, unsigned char)
Set the default colour pair for text (ANSI version).
- int `cucul_set_color_argb` (`cucul_canvas_t *`, unsigned int, unsigned int)
Set the default colour pair for text (truecolor version).
- int `cucul_printf` (`cucul_canvas_t *`, int, int, char const *,...)
Print a formatted string.
- int `cucul_clear_canvas` (`cucul_canvas_t *`)
Clear the canvas.
- int `cucul_set_canvas_handle` (`cucul_canvas_t *`, int, int)
Set cursor handle.
- int `cucul_get_canvas_handle_x` (`cucul_canvas_t` const *)
Get X handle position.
- int `cucul_get_canvas_handle_y` (`cucul_canvas_t` const *)
Get Y handle position.
- int `cucul_blit` (`cucul_canvas_t *`, int, int, `cucul_canvas_t` const *, `cucul_canvas_t` const *)
Blit a canvas onto another one.
- int `cucul_set_canvas_boundaries` (`cucul_canvas_t *`, int, int, unsigned int, unsigned int)
Set a canvas' new boundaries.
- int `cucul_invert` (`cucul_canvas_t *`)
Invert a canvas' colours.
- int `cucul_flip` (`cucul_canvas_t *`)
Flip a canvas horizontally.
- int `cucul_flop` (`cucul_canvas_t *`)
Flip a canvas vertically.
- int `cucul_rotate_180` (`cucul_canvas_t *`)
Rotate a canvas.
- int `cucul_rotate_left` (`cucul_canvas_t *`)
Rotate a canvas, 90 degrees counterclockwise.
- int `cucul_rotate_right` (`cucul_canvas_t *`)
Rotate a canvas, 90 degrees counterclockwise.
- int `cucul_stretch_left` (`cucul_canvas_t *`)
Rotate and stretch a canvas, 90 degrees counterclockwise.
- int `cucul_stretch_right` (`cucul_canvas_t *`)

Rotate and stretch a canvas, 90 degrees clockwise.

- unsigned char [cucul_attr_to_ansi](#) (unsigned long int)
Get DOS ANSI information from attribute.
- unsigned char [cucul_attr_to_ansi_fg](#) (unsigned long int)
Get ANSI foreground information from attribute.
- unsigned char [cucul_attr_to_ansi_bg](#) (unsigned long int)
Get ANSI background information from attribute.
- unsigned int [cucul_attr_to_rgb12_fg](#) (unsigned long int)
Get 12-bit RGB foreground information from attribute.
- unsigned int [cucul_attr_to_rgb12_bg](#) (unsigned long int)
Get 12-bit RGB background information from attribute.
- void [cucul_attr_to_argb64](#) (unsigned long int, unsigned char[8])
Get 64-bit ARGB information from attribute.
- unsigned long int [cucul_utf8_to_utf32](#) (char const *, unsigned int *)
Convert a UTF-8 character to UTF-32.
- unsigned int [cucul_utf32_to_utf8](#) (char *, unsigned long int)
Convert a UTF-32 character to UTF-8.
- unsigned char [cucul_utf32_to_cp437](#) (unsigned long int)
Convert a UTF-32 character to CP437.
- unsigned long int [cucul_cp437_to_utf32](#) (unsigned char)
Convert a CP437 character to UTF-32.
- char [cucul_utf32_to_ascii](#) (unsigned long int)
Convert a UTF-32 character to ASCII.
- int [cucul_utf32_is_fullwidth](#) (unsigned long int)
Tell whether a UTF-32 character is fullwidth.
- int [cucul_draw_line](#) (cucul_canvas_t *, int, int, int, int, unsigned long int)
Draw a line on the canvas using the given character.
- int [cucul_draw_polyline](#) (cucul_canvas_t *, int const x[], int const y[], int, unsigned long int)
Draw a polyline.
- int [cucul_draw_thin_line](#) (cucul_canvas_t *, int, int, int, int)
Draw a thin line on the canvas, using ASCII art.
- int [cucul_draw_thin_polyline](#) (cucul_canvas_t *, int const x[], int const y[], int)
Draw an ASCII art thin polyline.

- int [cucul_draw_circle](#) (cucul_canvas_t *, int, int, int, unsigned long int)
Draw a circle on the canvas using the given character.
- int [cucul_draw_ellipse](#) (cucul_canvas_t *, int, int, int, int, unsigned long int)
Draw an ellipse on the canvas using the given character.
- int [cucul_draw_thin_ellipse](#) (cucul_canvas_t *, int, int, int, int)
Draw a thin ellipse on the canvas.
- int [cucul_fill_ellipse](#) (cucul_canvas_t *, int, int, int, int, unsigned long int)
Fill an ellipse on the canvas using the given character.
- int [cucul_draw_box](#) (cucul_canvas_t *, int, int, int, int, unsigned long int)
Draw a box on the canvas using the given character.
- int [cucul_draw_thin_box](#) (cucul_canvas_t *, int, int, int, int)
Draw a thin box on the canvas.
- int [cucul_draw_cp437_box](#) (cucul_canvas_t *, int, int, int, int)
Draw a box on the canvas using CP437 characters.
- int [cucul_fill_box](#) (cucul_canvas_t *, int, int, int, int, unsigned long int)
Fill a box on the canvas using the given character.
- int [cucul_draw_triangle](#) (cucul_canvas_t *, int, int, int, int, int, int, unsigned long int)
Draw a triangle on the canvas using the given character.
- int [cucul_draw_thin_triangle](#) (cucul_canvas_t *, int, int, int, int, int, int)
Draw a thin triangle on the canvas.
- int [cucul_fill_triangle](#) (cucul_canvas_t *, int, int, int, int, int, int, unsigned long int)
Fill a triangle on the canvas using the given character.
- unsigned int [cucul_get_frame_count](#) (cucul_canvas_t const *)
Get the number of frames in a canvas.
- int [cucul_set_frame](#) (cucul_canvas_t *, unsigned int)
Activate a given canvas frame.
- char const * [cucul_get_frame_name](#) (cucul_canvas_t const *)
Get the current frame's name.
- int [cucul_set_frame_name](#) (cucul_canvas_t *, char const *)
Set the current frame's name.
- int [cucul_create_frame](#) (cucul_canvas_t *, unsigned int)
Add a frame to a canvas.
- int [cucul_free_frame](#) (cucul_canvas_t *, unsigned int)
Remove a frame from a canvas.

- `cucul_dither_t * cucul_create_dither` (unsigned int, unsigned int, unsigned int, unsigned int, unsigned long int, unsigned long int, unsigned long int, unsigned long int)
Create an internal dither object.
- `int cucul_set_dither_palette` (`cucul_dither_t *`, unsigned int r[], unsigned int g[], unsigned int b[], unsigned int a[])
Set the palette of an 8bpp dither object.
- `int cucul_set_dither_brightness` (`cucul_dither_t *`, float)
Set the brightness of a dither object.
- `float cucul_get_dither_brightness` (`cucul_dither_t const *`)
Get the brightness of a dither object.
- `int cucul_set_dither_gamma` (`cucul_dither_t *`, float)
Set the gamma of a dither object.
- `float cucul_get_dither_gamma` (`cucul_dither_t const *`)
Get the gamma of a dither object.
- `int cucul_set_dither_contrast` (`cucul_dither_t *`, float)
Set the contrast of a dither object.
- `float cucul_get_dither_contrast` (`cucul_dither_t const *`)
Get the contrast of a dither object.
- `int cucul_set_dither_antialias` (`cucul_dither_t *`, char const *)
Set dither antialiasing.
- `char const *const * cucul_get_dither_antialias_list` (`cucul_dither_t const *`)
Get available antialiasing methods.
- `char const * cucul_get_dither_antialias` (`cucul_dither_t const *`)
Get current antialiasing method.
- `int cucul_set_dither_color` (`cucul_dither_t *`, char const *)
Choose colours used for dithering.
- `char const *const * cucul_get_dither_color_list` (`cucul_dither_t const *`)
Get available colour modes.
- `char const * cucul_get_dither_color` (`cucul_dither_t const *`)
Get current colour mode.
- `int cucul_set_dither_charset` (`cucul_dither_t *`, char const *)
Choose characters used for dithering.
- `char const *const * cucul_get_dither_charset_list` (`cucul_dither_t const *`)
Get available dither character sets.

- char const * [cucul_get_dither_charset](#) (cucul_dither_t const *)
Get current character set.
- int [cucul_set_dither_algorithm](#) (cucul_dither_t *, char const *)
Set dithering algorithm.
- char const *const * [cucul_get_dither_algorithm_list](#) (cucul_dither_t const *)
Get dithering algorithms.
- char const * [cucul_get_dither_algorithm](#) (cucul_dither_t const *)
Get current dithering algorithm.
- int [cucul_dither_bitmap](#) (cucul_canvas_t *, int, int, int, int, cucul_dither_t const *, void *)
Dither a bitmap on the canvas.
- int [cucul_free_dither](#) (cucul_dither_t *)
Free the memory associated with a dither.
- cucul_font_t * [cucul_load_font](#) (void const *, unsigned int)
Load a font from memory for future use.
- char const *const * [cucul_get_font_list](#) (void)
Get available builtin fonts.
- unsigned int [cucul_get_font_width](#) (cucul_font_t const *)
Get a font's standard glyph width.
- unsigned int [cucul_get_font_height](#) (cucul_font_t const *)
Get a font's standard glyph height.
- unsigned long int const * [cucul_get_font_blocks](#) (cucul_font_t const *)
Get a font's list of supported glyphs.
- int [cucul_render_canvas](#) (cucul_canvas_t const *, cucul_font_t const *, void *, unsigned int, unsigned int, unsigned int)
Render the canvas onto an image buffer.
- int [cucul_free_font](#) (cucul_font_t *)
Free a font structure.
- long int [cucul_import_memory](#) (cucul_canvas_t *, void const *, unsigned long int, char const *)
Import a memory buffer into a canvas.
- long int [cucul_import_file](#) (cucul_canvas_t *, char const *, char const *)
Import a file into a canvas.
- char const *const * [cucul_get_import_list](#) (void)
Get available import formats.

- void * [cucul_export_memory](#) ([cucul_canvas_t](#) const *, char const *, unsigned long int *)
Export a canvas into a foreign format.
- char const *const * [cucul_get_export_list](#) (void)
Get available export formats.

4.2.1 Detailed Description

Version:

\$Id\$

Author:

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

4.2.2 Define Documentation

4.2.2.1 #define CUCUL_API_VERSION_1

libcucul API version

4.2.3 Typedef Documentation

4.2.3.1 typedef struct cucul_canvas cucul_canvas_t

libcucul canvas

4.2.3.2 typedef struct cucul_dither cucul_dither_t

dither structure

4.2.3.3 typedef struct cucul_font cucul_font_t

font structure

Index

caca.h, [61](#)

CACA_EVENT_ANY, [64](#)
CACA_EVENT_KEY_PRESS, [64](#)
CACA_EVENT_KEY_RELEASE, [64](#)
CACA_EVENT_MOUSE_MOTION, [64](#)
CACA_EVENT_MOUSE_PRESS, [64](#)
CACA_EVENT_MOUSE_RELEASE, [64](#)
CACA_EVENT_NONE, [64](#)
CACA_EVENT_QUIT, [64](#)
CACA_EVENT_RESIZE, [64](#)
CACA_KEY_BACKSPACE, [65](#)
CACA_KEY_CTRL_A, [65](#)
CACA_KEY_CTRL_B, [65](#)
CACA_KEY_CTRL_C, [65](#)
CACA_KEY_CTRL_D, [65](#)
CACA_KEY_CTRL_E, [65](#)
CACA_KEY_CTRL_F, [65](#)
CACA_KEY_CTRL_G, [65](#)
CACA_KEY_CTRL_J, [65](#)
CACA_KEY_CTRL_K, [65](#)
CACA_KEY_CTRL_L, [65](#)
CACA_KEY_CTRL_N, [65](#)
CACA_KEY_CTRL_O, [65](#)
CACA_KEY_CTRL_P, [65](#)
CACA_KEY_CTRL_Q, [65](#)
CACA_KEY_CTRL_R, [65](#)
CACA_KEY_CTRL_T, [65](#)
CACA_KEY_CTRL_U, [65](#)
CACA_KEY_CTRL_V, [65](#)
CACA_KEY_CTRL_W, [65](#)
CACA_KEY_CTRL_X, [65](#)
CACA_KEY_CTRL_Y, [65](#)
CACA_KEY_CTRL_Z, [65](#)
CACA_KEY_DELETE, [65](#)
CACA_KEY_DOWN, [65](#)
CACA_KEY_END, [65](#)
CACA_KEY_ESCAPE, [65](#)
CACA_KEY_F1, [66](#)
CACA_KEY_F10, [66](#)
CACA_KEY_F11, [66](#)
CACA_KEY_F12, [66](#)
CACA_KEY_F13, [66](#)
CACA_KEY_F14, [66](#)
CACA_KEY_F15, [66](#)
CACA_KEY_F2, [66](#)
CACA_KEY_F3, [66](#)
CACA_KEY_F4, [66](#)
CACA_KEY_F5, [66](#)
CACA_KEY_F6, [66](#)
CACA_KEY_F7, [66](#)

CACA_KEY_F8, [66](#)
CACA_KEY_F9, [66](#)
CACA_KEY_HOME, [65](#)
CACA_KEY_INSERT, [65](#)
CACA_KEY_LEFT, [65](#)
CACA_KEY_PAGEDOWN, [65](#)
CACA_KEY_PAGEUP, [65](#)
CACA_KEY_PAUSE, [65](#)
CACA_KEY_RETURN, [65](#)
CACA_KEY_RIGHT, [65](#)
CACA_KEY_TAB, [65](#)
CACA_KEY_UNKNOWN, [65](#)
CACA_KEY_UP, [65](#)
CACA_API_VERSION_1, [64](#)
caca_display_t, [64](#)
caca_event_t, [64](#)
caca_event_type, [64](#)
caca_key, [64](#)
CACA_EVENT_ANY
 caca.h, [64](#)
CACA_EVENT_KEY_PRESS
 caca.h, [64](#)
CACA_EVENT_KEY_RELEASE
 caca.h, [64](#)
CACA_EVENT_MOUSE_MOTION
 caca.h, [64](#)
CACA_EVENT_MOUSE_PRESS
 caca.h, [64](#)
CACA_EVENT_MOUSE_RELEASE
 caca.h, [64](#)
CACA_EVENT_NONE
 caca.h, [64](#)
CACA_EVENT_QUIT
 caca.h, [64](#)
CACA_EVENT_RESIZE
 caca.h, [64](#)
CACA_KEY_BACKSPACE
 caca.h, [65](#)
CACA_KEY_CTRL_A
 caca.h, [65](#)
CACA_KEY_CTRL_B
 caca.h, [65](#)
CACA_KEY_CTRL_C
 caca.h, [65](#)
CACA_KEY_CTRL_D
 caca.h, [65](#)
CACA_KEY_CTRL_E
 caca.h, [65](#)
CACA_KEY_CTRL_F
 caca.h, [65](#)

- CACA_KEY_CTRL_G
 - caca.h, [65](#)
- CACA_KEY_CTRL_J
 - caca.h, [65](#)
- CACA_KEY_CTRL_K
 - caca.h, [65](#)
- CACA_KEY_CTRL_L
 - caca.h, [65](#)
- CACA_KEY_CTRL_N
 - caca.h, [65](#)
- CACA_KEY_CTRL_O
 - caca.h, [65](#)
- CACA_KEY_CTRL_P
 - caca.h, [65](#)
- CACA_KEY_CTRL_Q
 - caca.h, [65](#)
- CACA_KEY_CTRL_R
 - caca.h, [65](#)
- CACA_KEY_CTRL_T
 - caca.h, [65](#)
- CACA_KEY_CTRL_U
 - caca.h, [65](#)
- CACA_KEY_CTRL_V
 - caca.h, [65](#)
- CACA_KEY_CTRL_W
 - caca.h, [65](#)
- CACA_KEY_CTRL_X
 - caca.h, [65](#)
- CACA_KEY_CTRL_Y
 - caca.h, [65](#)
- CACA_KEY_CTRL_Z
 - caca.h, [65](#)
- CACA_KEY_DELETE
 - caca.h, [65](#)
- CACA_KEY_DOWN
 - caca.h, [65](#)
- CACA_KEY_END
 - caca.h, [65](#)
- CACA_KEY_ESCAPE
 - caca.h, [65](#)
- CACA_KEY_F1
 - caca.h, [66](#)
- CACA_KEY_F10
 - caca.h, [66](#)
- CACA_KEY_F11
 - caca.h, [66](#)
- CACA_KEY_F12
 - caca.h, [66](#)
- CACA_KEY_F13
 - caca.h, [66](#)
- CACA_KEY_F14
 - caca.h, [66](#)
- CACA_KEY_F15
 - caca.h, [66](#)
- CACA_KEY_F2
 - caca.h, [66](#)
- CACA_KEY_F3
 - caca.h, [66](#)
- CACA_KEY_F4
 - caca.h, [66](#)
- CACA_KEY_F5
 - caca.h, [66](#)
- CACA_KEY_F6
 - caca.h, [66](#)
- CACA_KEY_F7
 - caca.h, [66](#)
- CACA_KEY_F8
 - caca.h, [66](#)
- CACA_KEY_F9
 - caca.h, [66](#)
- CACA_KEY_HOME
 - caca.h, [65](#)
- CACA_KEY_INSERT
 - caca.h, [65](#)
- CACA_KEY_LEFT
 - caca.h, [65](#)
- CACA_KEY_PAGEDOWN
 - caca.h, [65](#)
- CACA_KEY_PAGEUP
 - caca.h, [65](#)
- CACA_KEY_PAUSE
 - caca.h, [65](#)
- CACA_KEY_RETURN
 - caca.h, [65](#)
- CACA_KEY_RIGHT
 - caca.h, [65](#)
- CACA_KEY_TAB
 - caca.h, [65](#)
- CACA_KEY_UNKNOWN
 - caca.h, [65](#)
- CACA_KEY_UP
 - caca.h, [65](#)
- CACA_API_VERSION_1
 - caca.h, [64](#)
- caca_create_display
 - libcaca, [52](#)
- caca_display_t
 - caca.h, [64](#)
- caca_event, [60](#)
 - caca_get_event, [57](#)
 - caca_get_event_key_ch, [58](#)
 - caca_get_event_key_utf32, [58](#)
 - caca_get_event_key_utf8, [59](#)
 - caca_get_event_mouse_button, [59](#)
 - caca_get_event_mouse_x, [59](#)
 - caca_get_event_mouse_y, [60](#)
 - caca_get_event_resize_height, [60](#)
 - caca_get_event_resize_width, [60](#)

- caca_get_event_type, 58
- caca_get_mouse_x, 57
- caca_get_mouse_y, 57
- caca_event_t
 - caca.h, 64
- caca_event_type
 - caca.h, 64
- caca_free_display
 - libcaca, 52
- caca_get_canvas
 - libcaca, 53
- caca_get_display_height
 - libcaca, 54
- caca_get_display_time
 - libcaca, 53
- caca_get_display_width
 - libcaca, 54
- caca_get_event
 - caca_event, 57
- caca_get_event_key_ch
 - caca_event, 58
- caca_get_event_key_utf32
 - caca_event, 58
- caca_get_event_key_utf8
 - caca_event, 59
- caca_get_event_mouse_button
 - caca_event, 59
- caca_get_event_mouse_x
 - caca_event, 59
- caca_get_event_mouse_y
 - caca_event, 60
- caca_get_event_resize_height
 - caca_event, 60
- caca_get_event_resize_width
 - caca_event, 60
- caca_get_event_type
 - caca_event, 58
- caca_get_mouse_x
 - caca_event, 57
- caca_get_mouse_y
 - caca_event, 57
- caca_get_version
 - libcaca, 55
- caca_key
 - caca.h, 64
- caca_refresh_display
 - libcaca, 53
- caca_set_cursor
 - libcaca, 55
- caca_set_display_time
 - libcaca, 53
- caca_set_display_title
 - libcaca, 54
- caca_set_mouse
 - libcaca, 55
- cucul.h, 66
 - CUCUL_API_VERSION_1, 73
 - cucul_canvas_t, 73
 - cucul_dither_t, 73
 - cucul_font_t, 73
- CUCUL_API_VERSION_1
 - cucul.h, 73
- cucul_attr
 - CUCUL_BLACK, 3
 - CUCUL_BLINK, 4
 - CUCUL_BLUE, 3
 - CUCUL_BOLD, 4
 - CUCUL_BROWN, 3
 - CUCUL_CYAN, 3
 - CUCUL_DARKGRAY, 3
 - CUCUL_DEFAULT, 4
 - CUCUL_GREEN, 3
 - CUCUL_ITALICS, 4
 - CUCUL_LIGHTBLUE, 3
 - CUCUL_LIGHTCYAN, 3
 - CUCUL_LIGHTGRAY, 3
 - CUCUL_LIGHTGREEN, 3
 - CUCUL_LIGHTMAGENTA, 4
 - CUCUL_LIGHTRED, 3
 - CUCUL_MAGENTA, 3
 - CUCUL_RED, 3
 - CUCUL_TRANSPARENT, 4
 - CUCUL_UNDERLINE, 4
 - CUCUL_WHITE, 4
 - CUCUL_YELLOW, 4
- cucul_attr_to_ansi
 - cucul_attributes, 22
- cucul_attr_to_ansi_bg
 - cucul_attributes, 22
- cucul_attr_to_ansi_fg
 - cucul_attributes, 22
- cucul_attr_to_argb64
 - cucul_attributes, 23
- cucul_attr_to_rgb12_bg
 - cucul_attributes, 23
- cucul_attr_to_rgb12_fg
 - cucul_attributes, 23
- cucul_attributes
 - cucul_attr_to_ansi, 22
 - cucul_attr_to_ansi_bg, 22
 - cucul_attr_to_ansi_fg, 22
 - cucul_attr_to_argb64, 23
 - cucul_attr_to_rgb12_bg, 23
 - cucul_attr_to_rgb12_fg, 23
- CUCUL_BLACK
 - cucul_attr, 3
- CUCUL_BLINK
 - cucul_attr, 4

- cucul_blit
 - cucul_canvas, 17
- CUCUL_BLUE
 - cucul_attr, 3
- CUCUL_BOLD
 - cucul_attr, 4
- CUCUL_BROWN
 - cucul_attr, 3
- cucul_canvas
 - cucul_blit, 17
 - cucul_clear_canvas, 16
 - cucul_get_attr, 13
 - cucul_get_canvas_handle_x, 16
 - cucul_get_canvas_handle_y, 16
 - cucul_get_char, 12
 - cucul_get_cursor_x, 11
 - cucul_get_cursor_y, 11
 - cucul_gotoxy, 11
 - CUCUL_MAGIC_FULLWIDTH, 10
 - cucul_printf, 15
 - cucul_put_attr, 14
 - cucul_put_char, 11
 - cucul_put_str, 12
 - cucul_set_attr, 13
 - cucul_set_canvas_boundaries, 17
 - cucul_set_canvas_handle, 16
 - cucul_set_color_ansi, 14
 - cucul_set_color_argb, 15
- cucul_canvas_t
 - cucul.h, 73
- cucul_charset
 - cucul_cp437_to_utf32, 25
 - cucul_utf32_is_fullwidth, 26
 - cucul_utf32_to_ascii, 26
 - cucul_utf32_to_cp437, 25
 - cucul_utf32_to_utf8, 25
 - cucul_utf8_to_utf32, 25
- cucul_clear_canvas
 - cucul_canvas, 16
- cucul_cp437_to_utf32
 - cucul_charset, 25
- cucul_create_canvas
 - libcucul, 5
- cucul_create_dither
 - cucul_dither, 38
- cucul_create_frame
 - cucul_frame, 35
- CUCUL_CYAN
 - cucul_attr, 3
- CUCUL_DARKGRAY
 - cucul_attr, 3
- CUCUL_DEFAULT
 - cucul_attr, 4
- cucul_dither
 - cucul_create_dither, 38
 - cucul_dither_bitmap, 44
 - cucul_free_dither, 44
 - cucul_get_dither_algorithm, 44
 - cucul_get_dither_algorithm_list, 43
 - cucul_get_dither_antialias, 41
 - cucul_get_dither_antialias_list, 40
 - cucul_get_dither_brightness, 39
 - cucul_get_dither_charset, 43
 - cucul_get_dither_charset_list, 42
 - cucul_get_dither_color, 42
 - cucul_get_dither_color_list, 41
 - cucul_get_dither_contrast, 40
 - cucul_get_dither_gamma, 39
 - cucul_set_dither_algorithm, 43
 - cucul_set_dither_antialias, 40
 - cucul_set_dither_brightness, 38
 - cucul_set_dither_charset, 42
 - cucul_set_dither_color, 41
 - cucul_set_dither_contrast, 39
 - cucul_set_dither_gamma, 39
 - cucul_set_dither_palette, 38
- cucul_dither_bitmap
 - cucul_dither, 44
- cucul_dither_t
 - cucul.h, 73
- cucul_draw_box
 - cucul_primitives, 30
- cucul_draw_circle
 - cucul_primitives, 29
- cucul_draw_cp437_box
 - cucul_primitives, 31
- cucul_draw_ellipse
 - cucul_primitives, 29
- cucul_draw_line
 - cucul_primitives, 28
- cucul_draw_polyline
 - cucul_primitives, 28
- cucul_draw_thin_box
 - cucul_primitives, 31
- cucul_draw_thin_ellipse
 - cucul_primitives, 30
- cucul_draw_thin_line
 - cucul_primitives, 28
- cucul_draw_thin_polyline
 - cucul_primitives, 29
- cucul_draw_thin_triangle
 - cucul_primitives, 32
- cucul_draw_triangle
 - cucul_primitives, 32
- cucul_export_memory
 - cucul_importexport, 50
- cucul_fill_box
 - cucul_primitives, 31

- cucul_fill_ellipse
 - cucul_primitives, 30
- cucul_fill_triangle
 - cucul_primitives, 33
- cucul_flip
 - cucul_transform, 19
- cucul_flop
 - cucul_transform, 19
- cucul_font
 - cucul_free_font, 47
 - cucul_get_font_blocks, 47
 - cucul_get_font_height, 46
 - cucul_get_font_list, 46
 - cucul_get_font_width, 46
 - cucul_load_font, 45
 - cucul_render_canvas, 47
- cucul_font_t
 - cucul.h, 73
- cucul_frame
 - cucul_create_frame, 35
 - cucul_free_frame, 35
 - cucul_get_frame_count, 34
 - cucul_get_frame_name, 34
 - cucul_set_frame, 34
 - cucul_set_frame_name, 35
- cucul_free_canvas
 - libcucul, 8
- cucul_free_dither
 - cucul_dither, 44
- cucul_free_font
 - cucul_font, 47
- cucul_free_frame
 - cucul_frame, 35
- cucul_get_attr
 - cucul_canvas, 13
- cucul_get_canvas_attrs
 - libcucul, 8
- cucul_get_canvas_chars
 - libcucul, 8
- cucul_get_canvas_handle_x
 - cucul_canvas, 16
- cucul_get_canvas_handle_y
 - cucul_canvas, 16
- cucul_get_canvas_height
 - libcucul, 7
- cucul_get_canvas_width
 - libcucul, 7
- cucul_get_char
 - cucul_canvas, 12
- cucul_get_cursor_x
 - cucul_canvas, 11
- cucul_get_cursor_y
 - cucul_canvas, 11
- cucul_get_dither_algorithm
 - cucul_dither, 44
- cucul_get_dither_algorithm_list
 - cucul_dither, 43
- cucul_get_dither_antialias
 - cucul_dither, 41
- cucul_get_dither_antialias_list
 - cucul_dither, 40
- cucul_get_dither_brightness
 - cucul_dither, 39
- cucul_get_dither_charset
 - cucul_dither, 43
- cucul_get_dither_charset_list
 - cucul_dither, 42
- cucul_get_dither_color
 - cucul_dither, 42
- cucul_get_dither_color_list
 - cucul_dither, 41
- cucul_get_dither_contrast
 - cucul_dither, 40
- cucul_get_dither_gamma
 - cucul_dither, 39
- cucul_get_export_list
 - cucul_importexport, 51
- cucul_get_font_blocks
 - cucul_font, 47
- cucul_get_font_height
 - cucul_font, 46
- cucul_get_font_list
 - cucul_font, 46
- cucul_get_font_width
 - cucul_font, 46
- cucul_get_frame_count
 - cucul_frame, 34
- cucul_get_frame_name
 - cucul_frame, 34
- cucul_get_import_list
 - cucul_importexport, 50
- cucul_get_version
 - libcucul, 9
- cucul_gotoxy
 - cucul_canvas, 11
- CUCUL_GREEN
 - cucul_attr, 3
- cucul_import_file
 - cucul_importexport, 49
- cucul_import_memory
 - cucul_importexport, 48
- cucul_importexport
 - cucul_export_memory, 50
 - cucul_get_export_list, 51
 - cucul_get_import_list, 50
 - cucul_import_file, 49
 - cucul_import_memory, 48
- cucul_invert

- cucul_transform, 18
- CUCUL_ITALICS
 - cucul_attr, 4
- CUCUL_LIGHTBLUE
 - cucul_attr, 3
- CUCUL_LIGHTCYAN
 - cucul_attr, 3
- CUCUL_LIGHTGRAY
 - cucul_attr, 3
- CUCUL_LIGHTGREEN
 - cucul_attr, 3
- CUCUL_LIGHTMAGENTA
 - cucul_attr, 4
- CUCUL_LIGHTRED
 - cucul_attr, 3
- cucul_load_font
 - cucul_font, 45
- CUCUL_MAGENTA
 - cucul_attr, 3
- CUCUL_MAGIC_FULLWIDTH
 - cucul_canvas, 10
- cucul_manage_canvas
 - libcucul, 6
- cucul_primitives
 - cucul_draw_box, 30
 - cucul_draw_circle, 29
 - cucul_draw_cp437_box, 31
 - cucul_draw_ellipse, 29
 - cucul_draw_line, 28
 - cucul_draw_polyline, 28
 - cucul_draw_thin_box, 31
 - cucul_draw_thin_ellipse, 30
 - cucul_draw_thin_line, 28
 - cucul_draw_thin_polyline, 29
 - cucul_draw_thin_triangle, 32
 - cucul_draw_triangle, 32
 - cucul_fill_box, 31
 - cucul_fill_ellipse, 30
 - cucul_fill_triangle, 33
- cucul_printf
 - cucul_canvas, 15
- cucul_put_attr
 - cucul_canvas, 14
- cucul_put_char
 - cucul_canvas, 11
- cucul_put_str
 - cucul_canvas, 12
- cucul_rand
 - libcucul, 8
- CUCUL_RED
 - cucul_attr, 3
- cucul_render_canvas
 - cucul_font, 47
- cucul_rotate_180
 - cucul_transform, 19
- cucul_rotate_left
 - cucul_transform, 19
- cucul_rotate_right
 - cucul_transform, 20
- cucul_set_attr
 - cucul_canvas, 13
- cucul_set_canvas_boundaries
 - cucul_canvas, 17
- cucul_set_canvas_handle
 - cucul_canvas, 16
- cucul_set_canvas_size
 - libcucul, 6
- cucul_set_color_ansi
 - cucul_canvas, 14
- cucul_set_color_argb
 - cucul_canvas, 15
- cucul_set_dither_algorithm
 - cucul_dither, 43
- cucul_set_dither_antialias
 - cucul_dither, 40
- cucul_set_dither_brightness
 - cucul_dither, 38
- cucul_set_dither_charset
 - cucul_dither, 42
- cucul_set_dither_color
 - cucul_dither, 41
- cucul_set_dither_contrast
 - cucul_dither, 39
- cucul_set_dither_gamma
 - cucul_dither, 39
- cucul_set_dither_palette
 - cucul_dither, 38
- cucul_set_frame
 - cucul_frame, 34
- cucul_set_frame_name
 - cucul_frame, 35
- cucul_stretch_left
 - cucul_transform, 20
- cucul_stretch_right
 - cucul_transform, 21
- cucul_transform
 - cucul_flip, 19
 - cucul_flop, 19
 - cucul_invert, 18
 - cucul_rotate_180, 19
 - cucul_rotate_left, 19
 - cucul_rotate_right, 20
 - cucul_stretch_left, 20
 - cucul_stretch_right, 21
- CUCUL_TRANSPARENT
 - cucul_attr, 4
- CUCUL_UNDERLINE
 - cucul_attr, 4

- cucul_unmanage_canvas
 - libcucul, [6](#)
- cucul_utf32_is_fullwidth
 - cucul_charset, [26](#)
- cucul_utf32_to_ascii
 - cucul_charset, [26](#)
- cucul_utf32_to_cp437
 - cucul_charset, [25](#)
- cucul_utf32_to_utf8
 - cucul_charset, [25](#)
- cucul_utf8_to_utf32
 - cucul_charset, [25](#)
- CUCUL_WHITE
 - cucul_attr, [4](#)
- CUCUL_YELLOW
 - cucul_attr, [4](#)
- libcaca
 - caca_create_display, [52](#)
 - caca_free_display, [52](#)
 - caca_get_canvas, [53](#)
 - caca_get_display_height, [54](#)
 - caca_get_display_time, [53](#)
 - caca_get_display_width, [54](#)
 - caca_get_version, [55](#)
 - caca_refresh_display, [53](#)
 - caca_set_cursor, [55](#)
 - caca_set_display_time, [53](#)
 - caca_set_display_title, [54](#)
 - caca_set_mouse, [55](#)
- libcaca basic functions, [51](#)
- libcaca event handling, [56](#)
- libcucul
 - cucul_create_canvas, [5](#)
 - cucul_free_canvas, [8](#)
 - cucul_get_canvas_attrs, [8](#)
 - cucul_get_canvas_chars, [8](#)
 - cucul_get_canvas_height, [7](#)
 - cucul_get_canvas_width, [7](#)
 - cucul_get_version, [9](#)
 - cucul_manage_canvas, [6](#)
 - cucul_rand, [8](#)
 - cucul_set_canvas_size, [6](#)
 - cucul_unmanage_canvas, [6](#)
- libcucul attribute conversions, [21](#)
- libcucul attribute definitions, [2](#)
- libcucul basic functions, [4](#)
- libcucul bitmap dithering, [36](#)
- libcucul canvas drawing, [9](#)
- libcucul canvas frame handling, [33](#)
- libcucul canvas transformation, [18](#)
- libcucul character set conversions, [24](#)
- libcucul font handling, [45](#)
- libcucul importers/exporters from/to various, [48](#)
- libcucul primitives drawing, [26](#)