libcaca Reference Manual

0.99.beta11

Generated by Doxygen 1.5.2

Fri Jun 1 18:01:31 2007

Contents

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

1.2 Developer's documentation

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

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)
- Coding style

1.3 User's documentation

- Environment variables

1.4 Misc

- News
- Authors
- Thanks
- TODO list

1.5 License

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

2 libcaca 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.

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.

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.

2.1.2.18 #define CUCUL_TRANSPARENT 0x20

The transparent colour.

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_set_canvas_size (cucul_canvas_t *, unsigned int, unsigned int)

Resize a canvas.

- unsigned int cucul_get_canvas_width (cucul_canvas_t *)
  Get the canvas width.
- unsigned int cucul_get_canvas_height (cucul_canvas_t *)
  Get the canvas height.
- int cucul_free_canvas (cucul_canvas_t *)
  Uninitialise libcucul.
- int cucul_rand (int, int)
  Generate a random integer within a range.

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.

2.2.2.2 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.

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.

2.2.2.3 unsigned int cucul_get_canvas_width (cucul_canvas_t * *cv*)

Return the current canvas' width, in character cells.

This function never fails.

Parameters:

cv A libcucul canvas

Returns:

The canvas width.

2.2.2.4 unsigned int cucul_get_canvas_height (cucul_canvas_t * *cv*)

Returns the current canvas' height, in character cells.

This function never fails.

Parameters:

cv A libcucul canvas

Returns:

The canvas height.

2.2.2.5 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.

2.2.2.6 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).

2.3 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 *)
  Get X cursor position.
- int cucul_get_cursor_y (cucul_canvas_t *)
  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 *, 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 *, 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 formated 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 *)
  Get X handle position.
- int cucul_get_canvas_handle_y (cucul_canvas_t *)
  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.

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 * *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 * *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.

2.3.3.5 unsigned long int cucul_get_char (cucul_canvas_t * *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. Otherise, 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.

2.3.3.7 unsigned long int cucul_get_attr (cucul_canvas_t * *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.

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.

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.

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.

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.

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 * *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 * *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.

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.

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 (cucul_canvas_t *)
  Rotate a canvas.

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.

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 (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.

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.

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.

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.

2.6 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.
- 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.

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 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.

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.

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 plyline 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.

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.

2.7.2.9 int cucul_draw_box (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 upper-left corner of the box.

y1 Y coordinate of the upper-left corner of the box.

x2 X coordinate of the lower-right corner of the box.

y2 Y coordinate of the lower-right corner of the box.

ch UTF-32 character to be used to draw the box.

Returns:

This function always returns 0.

2.7.2.10 int cucul_draw_thin_box (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 upper-left corner of the box.

y1 Y coordinate of the upper-left corner of the box.

x2 X coordinate of the lower-right corner of the box.

y2 Y coordinate of the lower-right corner of the box.

Returns:

This function always returns 0.

2.7.2.11 int cucul_draw_cp437_box (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 upper-left corner of the box.

y1 Y coordinate of the upper-left corner of the box.

x2 X coordinate of the lower-right corner of the box.

y2 Y coordinate of the lower-right corner of the box.

Returns:

This function always returns 0.

2.7.2.12 int cucul_fill_box (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 upper-left corner of the box.

y1 Y coordinate of the upper-left corner of the box.

x2 X coordinate of the lower-right corner of the box.

y2 Y coordinate of the lower-right corner of the box.

ch UTF-32 character to be used to draw the box.

Returns:

This function always returns 0.

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.

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.

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.

2.8 libcucul canvas frame handling

Functions

- unsigned int cucul_get_frame_count (cucul_canvas_t *)

  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 *)

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 * *cv*)

Return the current canvas' frame count.

This function never fails.

Parameters:

cv A libcucul canvas

Returns:

The frame count

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.

2.8.2.3 char const* cucul_get_frame_name (cucul_canvas_t * *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#xxxxxxxx`" 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.

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 the 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.

- int cucul_set_dither_gamma (cucul_dither_t *, float)

  Set the gamma of a dither object.

- int cucul_set_dither_contrast (cucul_dither_t *, float)

  Set the contrast of a dither object.

- int cucul_set_dither_invert (cucul_dither_t *, int)

  Invert colors of dither.

- 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.

- 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.

- 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.

- int cucul_set_dither_mode (cucul_dither_t *, char const *)

  Set dithering method.

- char const *const * cucul_get_dither_mode_list (cucul_dither_t const *)

  Get dithering methods.

- 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 (0xfff).

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 int cucul_set_dither_gamma (cucul_dither_t * *d*, float *gamma*)

Set the gamma of dither.

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.

2.9.2.5 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.

2.9.2.6 int cucul_set_dither_invert (cucul_dither_t * *d*, int *value*)

Invert colors of dither.

This function never fails.

Parameters:

d Dither object.

value 0 for normal behaviour, 1 for invert

Returns:

This function always returns 0.

2.9.2.7 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.

2.9.2.8 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.

2.9.2.9 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.

2.9.2.10 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.

2.9.2.11 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.

2.9.2.12 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.

2.9.2.13 int cucul_set_dither_mode (cucul_dither_t * *d*, char const * *str*)

Tell the renderer which dithering method 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 method that needs to be used for the dithering.

Returns:

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

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

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

This function never fails.

Parameters:

d Dither object.

Returns:

An array of strings.

2.9.2.15 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.

2.9.2.16 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 *)
  Get a font's standard glyph width.
- unsigned int cucul_get_font_height (cucul_font_t *)
  Get a font's standard glyph height.
- unsigned long int const * cucul_get_font_blocks (cucul_font_t *)

Get a font's list of supported glyphs.

- int cucul_render_canvas (cucul_canvas_t *, cucul_font_t *, 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.

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 * *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 * *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 * *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 * *cv*, cucul_font_t * *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.

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 formats

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 *, 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

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.

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.

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 * *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.

- 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 *)

Get the display's average rendering time.

- unsigned int caca_get_display_width (caca_display_t *)

  Get the display width.

- unsigned int caca_get_display_height (caca_display_t *)

  Get the display height.

- int caca_set_display_title (caca_display_t *, char const *)

  Set the display title.

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 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 cavas.

Returns:

The caca graphical context or NULL if an error occurred.

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.

This function never fails.

Parameters:

dp The libcaca graphical context.

Returns:

This function always returns 0.

2.12.2.3 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.

2.12.2.4 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.

2.12.2.5 unsigned int caca_get_display_time (caca_display_t * *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.

2.12.2.6 unsigned int caca_get_display_width (caca_display_t * *dp*)

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

This function never fails.

Parameters:

dp The libcaca display context.

Returns:

The display width.

2.12.2.7 unsigned int caca_get_display_height (caca_display_t * *dp*)

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

This function never fails.

Parameters:

dp The libcaca display context.

Returns:

The display height.

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

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

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

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

Parameters:

dp The libcaca display context.

title The desired display title.

Returns:

0 upon success, -1 if an error occurred.

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 *)
  Return the X mouse coordinate.
- unsigned int caca_get_mouse_y (caca_display_t *)
  Return the Y mouse coordinate.
- int caca_set_mouse (caca_display_t *, int)
  Show or hide the mouse pointer.

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 timeouted.

2.13.2.2 unsigned int caca_get_mouse_x (caca_display_t * *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.

2.13.2.3 unsigned int caca_get_mouse_y (caca_display_t * *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.

2.13.2.4 int caca_set_mouse (caca_display_t * *dp*, int *flag*)

Show or hide the mouse pointer, 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 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.

3 libcaca Data Structure Documentation

3.1 caca_event Struct Reference

Handling of user events.

Public Types

- enum caca_event_type {
  CACA_EVENT_NONE = 0x0000, CACA_EVENT_KEY_PRESS = 0x0001, CACA_EVENT_-KEY_RELEASE = 0x0002, CACA_EVENT_MOUSE_PRESS = 0x0004,
  CACA_EVENT_MOUSE_RELEASE = 0x0008, CACA_EVENT_MOUSE_MOTION = 0x0010, CACA_EVENT_RESIZE = 0x0020, CACA_EVENT_QUIT = 0x0040,
  CACA_EVENT_ANY = 0xffff }
  User event type enumeration.

Data Fields

- enum caca_event::caca_event_type type

  User event type enumeration.User event type member.

- union {
  struct {
  unsigned int **x**
  unsigned int **y**
  unsigned int **button**
  } **mouse**
  struct {
  unsigned int **w**
  unsigned int **h**
  } **resize**
  struct {
  unsigned int **ch**
  unsigned long int **utf32**
  char **utf8** [8]
  } **key**
  } data

  User event data member.

3.1.1 Detailed Description

This structure is filled by caca_get_event() when an event is received. The *type* field is always valid.

3.1.2 Member Enumeration Documentation

3.1.2.1 enum caca_event::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.

3.1.3 Field Documentation

3.1.3.1 enum caca_event::caca_event_type caca_event::type

This field is always valid when caca_get_event() returns.

3.1.3.2 union { ... } caca_event::data

The validity of the *data* union depends on the value of the *type* field:

- `CACA_EVENT_NONE`: no field is valid.
- `CACA_EVENT_KEY_PRESS`, `CACA_EVENT_KEY_RELEASE`: the *data.key.ch* field is valid and contains either the ASCII value for the key, or an *enum caca_key* value. If the value is a printable ASCII character, the *data.key.utf32* and *data.key.utf8* fields are also filled and contain respectively the UTF-32/UCS-4 and the UTF-8 representations of the character. Otherwise, their content is undefined.
- `CACA_EVENT_MOUSE_PRESS`, `CACA_EVENT_MOUSE_RELEASE`: the *data.mouse.button* field is valid and contains the index of the mouse button that was pressed.
- `CACA_EVENT_MOUSE_MOTION`: the *data.mouse.x* and *data.mouse.y* fields are valid and contain the mouse coordinates in character cells.
- `CACA_EVENT_RESIZE`: the *data.resize.w* and *data.resize.h* fields are valid and contain the new width and height values of the *libcucul* canvas attached to *libcaca*.
- `CACA_EVENT_QUIT`: no other field is valid.

The result of accessing data members outside the above conditions is undefined.

4 libcaca File Documentation

4.1 caca.h File Reference

The *libcaca* public header.

Data Structures

- struct caca_event

  Handling of user events.

Defines

- #define CACA_API_VERSION_1

Typedefs

- typedef caca_display caca_display_t
- typedef caca_event caca_event_t

Enumerations

- enum caca_key {

  CACA_KEY_UNKNOWN = 0x00, CACA_KEY_CTRL_A = 0x01, CACA_KEY_CTRL_B = 0x02, CACA_KEY_CTRL_C = 0x03,

  CACA_KEY_CTRL_D = 0x04, CACA_KEY_CTRL_E = 0x05, CACA_KEY_CTRL_F = 0x06, CACA_KEY_CTRL_G = 0x07,

  CACA_KEY_BACKSPACE = 0x08, CACA_KEY_TAB = 0x09, CACA_KEY_CTRL_J = 0x0a, CACA_KEY_CTRL_K = 0x0b,

  CACA_KEY_CTRL_L = 0x0c, CACA_KEY_RETURN = 0x0d, CACA_KEY_CTRL_N = 0x0e, CACA_KEY_CTRL_O = 0x0f,

  CACA_KEY_CTRL_P = 0x10, CACA_KEY_CTRL_Q = 0x11, CACA_KEY_CTRL_R = 0x12, CACA_KEY_PAUSE = 0x13,

  CACA_KEY_CTRL_T = 0x14, CACA_KEY_CTRL_U = 0x15, CACA_KEY_CTRL_V = 0x16, CACA_KEY_CTRL_W = 0x17,

  CACA_KEY_CTRL_X = 0x18, CACA_KEY_CTRL_Y = 0x19, CACA_KEY_CTRL_Z = 0x1a, CACA_KEY_ESCAPE = 0x1b,

  CACA_KEY_DELETE = 0x7f, CACA_KEY_UP = 0x111, CACA_KEY_DOWN = 0x112, CACA_KEY_LEFT = 0x113,

  CACA_KEY_RIGHT = 0x114, CACA_KEY_INSERT = 0x115, CACA_KEY_HOME = 0x116, CACA_KEY_END = 0x117,

  CACA_KEY_PAGEUP = 0x118, CACA_KEY_PAGEDOWN = 0x119, CACA_KEY_F1 = 0x11a, CACA_KEY_F2 = 0x11b,

  CACA_KEY_F3 = 0x11c, CACA_KEY_F4 = 0x11d, CACA_KEY_F5 = 0x11e, CACA_KEY_F6 = 0x11f,

  CACA_KEY_F7 = 0x120, CACA_KEY_F8 = 0x121, CACA_KEY_F9 = 0x122, CACA_KEY_F10 = 0x123,

  CACA_KEY_F11 = 0x124, CACA_KEY_F12 = 0x125, CACA_KEY_F13 = 0x126, CACA_KEY_F14 = 0x127,

  CACA_KEY_F15 = 0x128 }

  Special key values.

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.

- 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 *)

Get the display's average rendering time.

- unsigned int caca_get_display_width (caca_display_t *)
  Get the display width.
- unsigned int caca_get_display_height (caca_display_t *)
  Get the display height.
- int caca_set_display_title (caca_display_t *, char const *)
  Set the display title.
- 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 *)
  Return the X mouse coordinate.
- unsigned int caca_get_mouse_y (caca_display_t *)
  Return the Y mouse coordinate.
- int caca_set_mouse (caca_display_t *, int)
  Show or hide the mouse pointer.

4.1.1 Detailed Description

Version:

$Id: caca.h 1059 2006-11-13 08:55:36Z sam $

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_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 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 cucul_canvas cucul_canvas_t
- typedef cucul_dither cucul_dither_t
- typedef cucul_font cucul_font_t

Functions

- cucul_canvas_t * cucul_create_canvas (unsigned int, unsigned int)
  Initialise a libcucul *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 *)
  Get the canvas width.
- unsigned int cucul_get_canvas_height (cucul_canvas_t *)
  Get the canvas height.
- int cucul_free_canvas (cucul_canvas_t *)
  Uninitialise libcucul.
- int cucul_rand (int, int)
  Generate a random integer within a range.
- int cucul_gotoxy (cucul_canvas_t *, int, int)
  Set cursor position.
- int cucul_get_cursor_x (cucul_canvas_t *)
  Get X cursor position.
- int cucul_get_cursor_y (cucul_canvas_t *)
  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 *, 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 *, 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 formated 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 *)
  Get X handle position.
- int cucul_get_canvas_handle_y (cucul_canvas_t *)
  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 (cucul_canvas_t *)
  Rotate a canvas.
- 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 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.
- 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 *)

  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 *)

  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.

- int cucul_set_dither_gamma (cucul_dither_t *, float)

  Set the gamma of a dither object.

- int cucul_set_dither_contrast (cucul_dither_t *, float)

  Set the contrast of a dither object.

- int cucul_set_dither_invert (cucul_dither_t *, int)

  Invert colors of dither.

- 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.
- 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.
- 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.
- int cucul_set_dither_mode (cucul_dither_t *, char const *)
  Set dithering method.
- char const *const * cucul_get_dither_mode_list (cucul_dither_t const *)
  Get dithering methods.
- 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 *)
  Get a font's standard glyph width.
- unsigned int cucul_get_font_height (cucul_font_t *)
  Get a font's standard glyph height.
- unsigned long int const * cucul_get_font_blocks (cucul_font_t *)
  Get a font's list of supported glyphs.
- int cucul_render_canvas (cucul_canvas_t *, cucul_font_t *, 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 *, 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: cucul.h 1063 2006-11-13 23:16:35Z sam $

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

5 libcaca Page Documentation

5.1 Authors

Sam Hocevar <sam@zoy.org>

- main programmer

Jean-Yves Lamoureux <jylam@lnxscene.org>

- cacaball
- OpenGL driver
- Pypycaca Python wrapper
- exporters
- network driver

John Beppu <beppu@lbox.org>

- Term::Caca Perl wrapper

5.2 News

5.2.1 Changes between 0.9.beta11 and 0.99.beta10

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

5.2.2 Changes between 0.9.beta10 and 0.99.beta9

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

5.2.3 Changes between 0.9.beta9 and 0.99.beta8

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

5.2.4 Changes between 0.9.beta8 and 0.99.beta7

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

5.2.5 Changes between 0.9.beta7 and 0.99.beta6

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

5.2.6 Changes between 0.9.beta6 and 0.99.beta5

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

5.2.7 Changes between 0.9.beta5 and 0.99.beta4

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

5.2.8 Changes between 0.9.beta4 and 0.99.beta3

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

5.2.9 Changes between 0.9.beta3 and 0.99.beta2

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

5.2.10 Changes between 0.9.beta2 and 0.99.beta1

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

5.2.11 Changes between 0.9 and 0.99.beta1

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

5.2.12 Changes between 0.8 and 0.9

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

5.2.13 Changes between 0.7 and 0.8

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

5.2.14 Changes between 0.6 and 0.7

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

5.2.15 Changes between 0.5 and 0.6

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

5.2.16 Changes between 0.4 and 0.5

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

5.2.17 Changes between 0.3 and 0.4

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

5.2.18 Changes between 0.2 and 0.3

- antialiasing support
- dithering, antialiasing and background mode can now be selected at runtime or in the environment using the CACA_BACKGROUND, CACA_DITHERING and CACA_ANTIALIASING variables
- alpha channel support in cacaview
- BMP loading support in cacaview even if Imlib2 is not present
- cacafire, a libcaca port of aafire

5.2.19 Changes between 0.1 and 0.2

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

5.2.20 New in 0.1

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

5.3 Thanks

5.3.1 Bugfixes and improvements

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

5.3.2 Reused code

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

5.3.3 Porters and packagers

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

5.4 TODO list

5.4.1 libcucul

5.4.1.1 API-dependent stuff

- support for holes in canvas (eg. not full EOL)
- support for TAB
- allow to change the canvas size in a per-frame basis.
- export attribute parsing functions such as attr_to_ansi4fg etc.

5.4.1.2 API-independent stuff

- Brightness, contrast support for bitmaps (the functions are here, we just need to fill them)
- Error distribution dithering
- Add a random factor to the random ditherer. No need to change the API for that, we can just pass `"random:10"` instead of `"random"` to the cucul_set_bitmap_dithering() function.
- Implement the colour modes set in cucul_set_bitmap_color(). For the moment only `"full16"` and `"16"` are implemented.
- Fix the thin ellipse rendering (currently it's only |s and -s, we could make them smoother by using ' ` , etc).
- better mask support in cucul_blit()
- optimise exporters so that they do not allocate huge blocks of memory when they only need half of it.

5.4.2 libcaca

5.4.2.1 API-dependent stuff

- beep support

5.4.2.2 API-independent stuff

- Write a Linux console output
- Better keyboard driver in an X terminal, see http://groups.yahoo.com/group/zepp/message/381
- Unicode support for X11 (maybe through Xft)
- fix Unicode support for ncurses (look at the nano source, it gets it right)
- and Jylam wants a framebuffer output
- write sample code for a text edit widget with cursor support

5.4.3 Language bindings

5.4.3.1 Needed

- Fix Python
- Fix Perl
- C# (it's the next big thing, believe me)
- PHP (together with the HTML output it would allow for nice web applications)

5.4.3.2 Not that important

- Ruby
- Java

5.4.4 Kernel mode

- keyboard support
- printf/fprintf are missing
- Improve malloc/free so that we can reuse freed memory

5.4.5 Documentation

- Write a tutorial.
- Draw a nicer logo

5.4.6 Applications

5.4.7 cacaview

- File browser
- open ANSI files
- save in different formats

5.4.8 cacadraw

- Only a skeleton exists yet. A modern ANSI editor that can also do Unicode.

5.5 The libcaca canvas format (version 1)

All types are big endian.

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

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

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

   uint32_t frames;           // Frame count

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

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

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

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

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

5.6 The libcaca font format (version 1)

All types are big endian.

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

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

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

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

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

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

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

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

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

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

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

5.7 Migrating from libcaca 0.x to the 1.0 API

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

5.7.1 Overview

The most important changes in the 1.0 API of *libcaca* are the *libcaca* / *libcucul* split and the object-oriented design. See these two examples for a rough idea of what changed:

```
#include <caca.h>

/* libcaca program - 0.x API */
int main(void)
{
    /* Initialise libcaca */
    caca_init();
    /* Set window title */
    caca_set_window_title("Window");
    /* Choose drawing colours */
    caca_set_color(CACA_COLOR_BLACK,
                   CACA_COLOR_WHITE);
    /* Draw a string at (0, 0) */
    caca_putstr(0, 0, "Hello world!");
    /* Refresh display */
    caca_refresh();
    /* Wait for a key press event */
    caca_wait_event(CACA_EVENT_KEY_PRESS);
    /* Clean up library */
    caca_end();

    return 0;
}
```

```
#include <cucul.h>
#include <caca.h>

/* libcaca program - 1.0 API */
int main(void)
{
    /* Initialise libcaca */
    cucul_canvas_t *cv;
    caca_display_t *dp;
    cv = cucul_create_canvas(0, 0);
    dp = caca_create_display(cv);
    /* Set window title */
    caca_set_display_title(dp, "Window");
    /* Choose drawing colours */
    cucul_set_color_ansi(cv, CUCUL_BLACK,
                             CUCUL_WHITE);
    /* Draw a string at (0, 0) */
    cucul_put_str(cv, 0, 0, "Hello world!");
    /* Refresh display */
    caca_refresh_display();
    /* Wait for a key press event */
    caca_get_event(dp, CACA_EVENT_KEY_PRESS,
                   NULL, -1);
    /* Clean up library */
    caca_free_display(dp);
    cucul_free_canvas(cv);

    return 0;
}
```

Note the following important things:

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

5.7.2 Migration strategy

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

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

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

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

5.7.3 Function equivalence list

5.7.3.1 Basic functions

- **caca_init()**: use cucul_create_canvas() to create a *libcucul* canvas, followed by caca_create_display() to attach a *libcaca* display to it.
- **caca_set_delay()**: use caca_set_display_time().
- **caca_get_feature()**: deprecated.
- **caca_set_feature()**: deprecated, see cucul_set_dither_antialias(), cucul_set_dither_color() and cucul_set_dither_mode() instead.
- **caca_get_feature_name()**: deprecated, see cucul_get_dither_mode_list(), cucul_get_dither_antialias_list() and cucul_get_dither_color_list() instead.
- **caca_get_rendertime()**: use caca_get_display_time().
- **caca_get_width()**: use cucul_get_canvas_width().
- **caca_get_height()**: use cucul_get_canvas_height().
- **caca_set_window_title()**: use caca_set_display_title().
- **caca_get_window_width()**: use caca_get_display_width().
- **caca_get_window_height()**: use caca_get_display_height().
- **caca_refresh()**: use caca_refresh_display().
- **caca_end()**: use caca_free_display() to detach the *libcaca* display, followed by cucul_free_canvas() to free the underlying *libcucul* canvas.

5.7.3.2 Event handling

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

5.7.3.3 Character printing

- **caca_set_color()**: use cucul_set_color_ansi() or cucul_set_color_argb().
- **caca_get_fg_color()**: use cucul_get_attr().
- **caca_get_bg_color()**: use cucul_get_attr().
- **caca_get_color_name()**: this function is now deprecated due to major uselessness.
- **caca_putchar()**: use cucul_put_char().
- **caca_putstr()**: use cucul_put_str().
- **caca_printf()**: use cucul_printf().
- **caca_clear()**: use cucul_clear_canvas().

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

- **caca_draw_line()**: use cucul_draw_line().
- **caca_draw_polyline()**: use cucul_draw_polyline().
- **caca_draw_thin_line()**: use cucul_draw_thin_line().
- **caca_draw_thin_polyline()**: use cucul_draw_thin_polyline().
- **caca_draw_circle()**: use cucul_draw_circle().
- **caca_draw_ellipse()**: use cucul_draw_ellipse().
- **caca_draw_thin_ellipse()**: use cucul_draw_thin_ellipse().
- **caca_fill_ellipse()**: use cucul_fill_ellipse().
- **caca_draw_box()**: use cucul_draw_box().
- **caca_draw_thin_box()**: use cucul_draw_thin_box() or cucul_draw_cp437_box().
- **caca_fill_box()**: use cucul_fill_box().
- **caca_draw_triangle()**: use cucul_draw_triangle().
- **caca_draw_thin_triangle()**: use cucul_draw_thin_triangle().
- **caca_fill_triangle()**: use cucul_fill_triangle().

5.7.3.5 Mathematical functions

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

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

- **caca_load_sprite()**: use cucul_import_file().
- **caca_get_sprite_frames()**: use cucul_get_frame_count().
- **caca_get_sprite_width()**: use cucul_get_canvas_width().
- **caca_get_sprite_height()**: use cucul_get_canvas_height().
- **caca_get_sprite_dx()**: use cucul_get_canvas_handle_x().
- **caca_get_sprite_dy()**: use cucul_get_canvas_handle_y().
- **caca_draw_sprite()**: use cucul_set_frame() and cucul_blit().
- **caca_free_sprite()**: use cucul_free_canvas().

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

- **caca_create_bitmap()**: use cucul_create_dither().
- **caca_set_bitmap_palette()**: use cucul_set_dither_palette().
- **caca_draw_bitmap()**: use cucul_dither_bitmap().
- **caca_free_bitmap()**: use cucul_free_dither().

5.7.4 Compilation

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

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

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

5.8 Coding style

5.8.1 General guidelines

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

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

5.8.2 C coding style

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

```
#define BROKEN 0
#define MAX(x, y) ((x > y) ? (x) : (y))

unsigned int x, y, w, h;
char *font_name;
void frobulate_every_three_seconds(void);
```

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

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

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

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

```
int function(int);

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

Do not put parentheses around return values:

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

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

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

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

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

5.8.3 C++ coding style

Nothing here yet.

5.9 A libcucul and libcaca tutorial

Before writing your first libcaca application, you need to know the difference between libcucul and libcaca :

- libcucul is the text rendering library. It will do all the work you actually need. From imports (text, ANSI, caca internal format, all of this supporting n-bytes unicode), to exports (sames formats, adding SVG, PostScript, TGA, HTML (both 3 and 4), IRC), it'll cover all your needs.
- libcaca handle everything that can be hardware related. It includes display (RAW, X11, OpenGL, Windows (GDI), conio (DOS), ncurses, slang, text VGA (IMB-Compatible)), keyboard (same drivers but RAW), mouse (same drivers but RAW and VGA), time and resize events (on windowed drivers).

So, you can write a libcucul only program, but you **can't** write a libcaca only program, it'll be nonsense. Period.

First, a working program, very simple, to check you can compile and run it :

```
#include <cucul.h>
#include <caca.h>

int main(void)
{
    /* Initialise libcaca */
    cucul_canvas_t *cv; caca_display_t *dp; caca_event_t ev;
    cv = cucul_create_canvas(0, 0);
    dp = caca_create_display(cv);
    /* Set window title */
    caca_set_display_title(dp, "Hello!");
    /* Choose drawing colours */
    cucul_set_color_ansi(cv, CUCUL_BLACK, CUCUL_WHITE);
    /* Draw a string at coordinates (0, 0) */
    cucul_putstr(cv, 0, 0, "This is a message");
    /* Refresh display */
    caca_refresh_display(dp);
    /* Wait for a key press event */
    caca_get_event(dp, CACA_EVENT_KEY_PRESS, &ev, -1);
    /* Clean up library */
    caca_free_display(dp);
    cucul_free_canvas(cv);

    return 0;
}
```

What does it do ? (we skip variable definitions, guessing you have a brain) :

- Create a cucul canvas. A canvas is where everything happens. Writing characters, sprites, strings, images, everything. It is mandatory and is the reason of libcuculs' beeing. Size is there a width of 0 pixels, and a height of 0 pixels. It'll be resized according to contents you put in it.
- Create a caca display. This is basically the window. Physically it can be a window (most of the displays), a console (ncurses, slang) or a real display (VGA).
- Set the window name of our display (only available in windowed displays, does nothing otherwise). (so this is libcaca related)

- Set current colors to black background, and white foreground of our canvas (so this is libcucul related)
- Put a string "This is a message" with current colors in our libcucul canvas.
- Refresh our caca display, whish was firstly attached to our canvas
- Wait for an event of type "CACA_EVENT_KEY_PRESS", which seems obvious.
- Free display (release memory)
- Free canvas (release memory and close window if any)

You can then compile this code under UNIX-like systems with following command : (you'll need pkg-config and gcc)

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

5.10 Environment variables

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

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

Index