# libcaca Reference Manual

0.99.beta1

Generated by Doxygen 1.4.6

Sun Apr 23 00:15:09 2006

# 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 cucul.h and caca.h headers.

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:

- 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 colour definitions

**Defines**

- #define CUCUL_COLOR_BLACK 0x00
- #define CUCUL_COLOR_BLUE 0x01
- #define CUCUL_COLOR_GREEN 0x02
- #define CUCUL_COLOR_CYAN 0x03
- #define CUCUL_COLOR_RED 0x04
- #define CUCUL_COLOR_MAGENTA 0x05
- #define CUCUL_COLOR_BROWN 0x06
- #define CUCUL_COLOR_LIGHTGRAY 0x07
- #define CUCUL_COLOR_DARKGRAY 0x08
- #define CUCUL_COLOR_LIGHTBLUE 0x09
- #define CUCUL_COLOR_LIGHTGREEN 0x0a
- #define CUCUL_COLOR_LIGHTCYAN 0x0b
- #define CUCUL_COLOR_LIGHTRED 0x0c
- #define CUCUL_COLOR_LIGHTMAGENTA 0x0d
- #define CUCUL_COLOR_YELLOW 0x0e
- #define CUCUL_COLOR_WHITE 0x0f
- #define CUCUL_COLOR_DEFAULT 0x10
- #define CUCUL_COLOR_TRANSPARENT 0x20

### 2.1.1 Detailed Description

Colours that can be used with cucul_set_color().

### 2.1.2 Define Documentation

#### 2.1.2.1 #define CUCUL_COLOR_BLACK 0x00

The colour index for black.

#### 2.1.2.2 #define CUCUL_COLOR_BLUE 0x01

The colour index for blue.

#### 2.1.2.3 #define CUCUL_COLOR_GREEN 0x02

The colour index for green.

#### 2.1.2.4 #define CUCUL_COLOR_CYAN 0x03

The colour index for cyan.

#### 2.1.2.5 #define CUCUL_COLOR_RED 0x04

The colour index for red.

#### 2.1.2.6 #define CUCUL_COLOR_MAGENTA 0x05

The colour index for magenta.

#### 2.1.2.7 #define CUCUL_COLOR_BROWN 0x06

The colour index for brown.

#### 2.1.2.8 #define CUCUL_COLOR_LIGHTGRAY 0x07

The colour index for light gray.

#### 2.1.2.9 #define CUCUL_COLOR_DARKGRAY 0x08

The colour index for dark gray.

#### 2.1.2.10 #define CUCUL_COLOR_LIGHTBLUE 0x09

The colour index for blue.

#### 2.1.2.11 #define CUCUL_COLOR_LIGHTGREEN 0x0a

The colour index for light green.

#### 2.1.2.12 #define CUCUL_COLOR_LIGHTCYAN 0x0b

The colour index for light cyan.

#### 2.1.2.13 #define CUCUL_COLOR_LIGHTRED 0x0c

The colour index for light red.

#### 2.1.2.14 #define CUCUL_COLOR_LIGHTMAGENTA 0x0d

The colour index for light magenta.

#### 2.1.2.15 #define CUCUL_COLOR_YELLOW 0x0e

The colour index for yellow.

#### 2.1.2.16 #define CUCUL_COLOR_WHITE 0x0f

The colour index for white.

#### 2.1.2.17 #define CUCUL_COLOR_DEFAULT 0x10

The output driver's default colour.

#### 2.1.2.18 #define CUCUL_COLOR_TRANSPARENT 0x20

The transparent colour.

## 2.2 libcucul basic functions

**Functions**

- cucul_canvas_t ∗ cucul_create_canvas (unsigned int, unsigned int)
  *Initialise a* libcucul *canvas.*
- void 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.*
- void 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*)

This function initialises 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.

If one of the desired canvas coordinates is zero, a default canvas size of 80x32 is used instead.

**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 void cucul_set_canvas_size (cucul_canvas_t $*$ *cv*, unsigned int *width*, unsigned int *height*)

This function sets 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.

**Parameters:**

***cv*** A libcucul canvas

***width*** The desired canvas width

***height*** The desired canvas height

#### 2.2.2.3 unsigned int cucul_get_canvas_width (cucul_canvas_t $*$ *cv*)

This function returns the current canvas width, in character cells.

**Parameters:**

***cv*** A libcucul canvas

**Returns:**

The canvas width.

#### 2.2.2.4 unsigned int cucul_get_canvas_height (cucul_canvas_t $*$ *cv*)

This function returns the current canvas height, in character cells.

**Parameters:**

***cv*** A libcucul canvas

**Returns:**

The canvas height.

#### 2.2.2.5 void cucul_free_canvas (cucul_canvas_t $*$ *cv*)

This function frees all resources allocated by cucul_create_canvas(). After cucul_free_canvas() has been called, no other *libcucul* functions may be used unless a new call to cucul_create_canvas() is done.

**Parameters:**

***cv*** A libcucul canvas

#### 2.2.2.6 int cucul_rand (int *min*, int *max*)

**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 buffer handling

### Functions

- unsigned long int cucul_get_buffer_size (cucul_buffer_t ∗)

  *Get the buffer size.*

- void ∗ cucul_get_buffer_data (cucul_buffer_t ∗)

  *Get the buffer data.*

- void cucul_free_buffer (cucul_buffer_t ∗)

  *Free a buffer.*

### 2.3.1 Detailed Description

These functions provide methods to handle libcucul buffers.

### 2.3.2 Function Documentation

#### 2.3.2.1 unsigned long int cucul_get_buffer_size (cucul_buffer_t ∗ *buf*)

This function returns the length (in bytes) of the memory area stored in the given *libcucul* buffer.

**Parameters:**

***buf*** A *libcucul* buffer

**Returns:**

The buffer data length.

#### 2.3.2.2 void∗ cucul_get_buffer_data (cucul_buffer_t ∗ *buf*)

This function returns a pointer to the memory area stored in the given *libcucul* buffer.

**Parameters:**

***buf*** A *libcucul* buffer

**Returns:**

A pointer to the buffer memory area.

#### 2.3.2.3 void cucul_free_buffer (cucul_buffer_t ∗ *buf*)

This function frees the structures associated with the given *libcucul* buffer.

**Parameters:**

***buf*** A *libcucul* buffer

## 2.4 libcucul canvas drawing

### Functions

- void cucul_set_color (cucul_canvas_t ∗, unsigned char, unsigned char)

  *Set the default colour pair.*

- void cucul_set_truecolor (cucul_canvas_t ∗, unsigned int, unsigned int)

  *Set the default colour pair (truecolor version).*

- char const ∗ cucul_get_color_name (unsigned int)

  *Translate a colour index into the colour's name.*

- void cucul_putchar (cucul_canvas_t ∗, int, int, char)

  *Print an ASCII character.*

- void cucul_putstr (cucul_canvas_t ∗, int, int, char const ∗)

  *Print a string.*

- void cucul_printf (cucul_canvas_t ∗, int, int, char const ∗,...)

  *Print a formated string.*

- void cucul_clear_canvas (cucul_canvas_t ∗)

  *Clear the canvas.*

- void cucul_blit (cucul_canvas_t ∗, int, int, cucul_canvas_t const ∗, cucul_canvas_t const ∗)

  *Blit a canvas onto another one.*

### 2.4.1 Detailed Description

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

### 2.4.2 Function Documentation

#### 2.4.2.1 void cucul_set_color (cucul_canvas_t ∗ *cv*, unsigned char *fg*, unsigned char *bg*)

This function sets the default ANSI colour pair. String functions such as caca_printf() and graphical primitive functions such as caca_draw_line() will use these colours.

Color values are those defined in cucul.h, such as CUCUL_COLOR_RED or CUCUL_COLOR_-TRANSPARENT.

**Parameters:**

***cv*** A handle to the libcucul canvas.

*fg* The requested foreground colour.

*bg* The requested background colour.

#### 2.4.2.2 void cucul_set_truecolor (cucul_canvas_t ∗ *cv*, unsigned int *fg*, unsigned int *bg*)

This function sets the default colour pair. String functions such as caca_printf() and graphical primitive functions such as caca_draw_line() will use these colours.

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).

**Parameters:**

*cv* A handle to the libcucul canvas.

*fg* The requested foreground colour.

*bg* The requested background colour.

#### 2.4.2.3 char const∗ cucul_get_color_name (unsigned int *color*)

This function translates a cucul_color enum into a human-readable description string of the associated colour.

**Parameters:**

*color* The colour value.

**Returns:**

A static string containing the colour's name.

#### 2.4.2.4 void cucul_putchar (cucul_canvas_t ∗ *cv*, int *x*, int *y*, char *ch*)

This function prints an ASCII character at the given coordinates, using the default foreground and background values. If the coordinates are outside the canvas boundaries, nothing is printed. If the character value is a non-printable character or is outside the ASCII range, it is replaced with a space. To print a sequence of bytes forming an UTF-8 character, use cucul_putstr() instead.

**Parameters:**

*cv* A handle to the libcucul canvas.

*x* X coordinate.

*y* Y coordinate.

*ch* The character to print.

#### 2.4.2.5 void cucul_putstr (cucul_canvas_t ∗ *cv*, int *x*, int *y*, char const ∗ *s*)

This function prints 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.

**Parameters:**

*cv* A handle to the libcucul canvas.

*x* X coordinate.

*y* Y coordinate.

*s* The string to print.

#### 2.4.2.6 void cucul_printf (cucul_canvas_t ∗ *cv*, int *x*, int *y*, char const ∗ *format*, ...)

This function formats 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.

**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.

#### 2.4.2.7 void cucul_clear_canvas (cucul_canvas_t ∗ *cv*)

This function clears the canvas using the current background colour.

**Parameters:**

***cv*** The canvas to clear.

#### 2.4.2.8 void cucul_blit (cucul_canvas_t ∗ *dst*, int *x*, int *y*, cucul_canvas_t const ∗ *src*, cucul_canvas_t const ∗ *mask*)

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

**Parameters:**

***dst*** The destination canvas.

***x*** X coordinate.

***y*** Y coordinate.

***src*** The source canvas.

***mask*** The mask canvas.

## 2.5 libcucul canvas transformation

### Functions

- void cucul_invert (cucul_canvas_t ∗)

  *Invert a canvas' colours.*

- void cucul_flip (cucul_canvas_t ∗)

  *Flip a canvas horizontally.*

- void cucul_flop (cucul_canvas_t ∗)

  *Flip a canvas vertically.*

- void cucul_rotate (cucul_canvas_t ∗)

  *Rotate a canvas.*

### 2.5.1 Detailed Description

These functions perform horizontal and vertical canvas flipping.

### 2.5.2 Function Documentation

#### 2.5.2.1 void cucul_invert (cucul_canvas_t ∗ *cv*)

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

**Parameters:**

*cv* The canvas to invert.

#### 2.5.2.2 void cucul_flip (cucul_canvas_t ∗ *cv*)

This function flips a canvas horizontally, choosing characters that look like the mirrored version wherever possible.

**Parameters:**

*cv* The canvas to flip.

#### 2.5.2.3 void cucul_flop (cucul_canvas_t ∗ *cv*)

This function flips a canvas vertically, choosing characters that look like the mirrored version wherever possible.

**Parameters:**

*cv* The canvas to flop.

#### 2.5.2.4 void cucul_rotate (cucul_canvas_t ∗ *cv*)

This function applies a 180 degrees transformation to a canvas, choosing characters that look like the mirrored version wherever possible.

**Parameters:**

*cv* The canvas to rotate.

## 2.6 libcucul primitives drawing

### Functions

- void cucul_draw_line (cucul_canvas_t ∗, int, int, int, int, char const ∗)

  *Draw a line on the canvas using the given character.*

- void cucul_draw_polyline (cucul_canvas_t ∗, int const x[ ], int const y[ ], int, char const ∗)

  *Draw a polyline.*

- void cucul_draw_thin_line (cucul_canvas_t ∗, int, int, int, int)

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

- void cucul_draw_thin_polyline (cucul_canvas_t ∗, int const x[ ], int const y[ ], int)

  *Draw an ASCII art thin polyline.*

- void cucul_draw_circle (cucul_canvas_t ∗, int, int, int, char const ∗)

  *Draw a circle on the canvas using the given character.*

- void cucul_draw_ellipse (cucul_canvas_t ∗, int, int, int, int, char const ∗)

  *Draw an ellipse on the canvas using the given character.*

- void cucul_draw_thin_ellipse (cucul_canvas_t ∗, int, int, int, int)

  *Draw a thin ellipse on the canvas.*

- void cucul_fill_ellipse (cucul_canvas_t ∗, int, int, int, int, char const ∗)

  *Fill an ellipse on the canvas using the given character.*

- void cucul_draw_box (cucul_canvas_t ∗, int, int, int, int, char const ∗)

  *Draw a box on the canvas using the given character.*

- void cucul_draw_thin_box (cucul_canvas_t ∗, int, int, int, int)

  *Draw a thin box on the canvas.*

- void cucul_fill_box (cucul_canvas_t ∗, int, int, int, int, char const ∗)

  *Fill a box on the canvas using the given character.*

- void cucul_draw_triangle (cucul_canvas_t ∗, int, int, int, int, int, int, char const ∗)

  *Draw a triangle on the canvas using the given character.*

- void cucul_draw_thin_triangle (cucul_canvas_t ∗, int, int, int, int, int, int)

  *Draw a thin triangle on the canvas.*

- void cucul_fill_triangle (cucul_canvas_t ∗, int, int, int, int, int, int, char const ∗)

  *Fill a triangle on the canvas using the given character.*

### 2.6.1 Detailed Description

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

### 2.6.2 Function Documentation

#### 2.6.2.1 void cucul_draw_line (cucul_canvas_t ∗ *cv*, int *x1*, int *y1*, int *x2*, int *y2*, char const ∗ *str*)

**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.

***str*** UTF-8 string containing the character to use to draw the line.

**Returns:**
void

#### 2.6.2.2 void cucul_draw_polyline (cucul_canvas_t ∗ *cv*, int const *x*[ ], int const *y*[ ], int *n*, char const ∗ *str*)

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.

**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.

***str*** UTF-8 string containing the character to use to draw the lines.

**Returns:**
void

#### 2.6.2.3 void cucul_draw_thin_line (cucul_canvas_t ∗ *cv*, int *x1*, int *y1*, int *x2*, int *y2*)

**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:**
void

#### 2.6.2.4 void 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.

**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:**
void

#### 2.6.2.5 void cucul_draw_circle (cucul_canvas_t ∗ *cv*, int *x*, int *y*, int *r*, char const ∗ *str*)

**Parameters:**

***cv*** The handle to the libcucul canvas.

***x*** Center X coordinate.

***y*** Center Y coordinate.

***r*** Circle radius.

***str*** UTF-8 string representing the character that should be used to draw the circle outline.

**Returns:**

void

#### 2.6.2.6 void cucul_draw_ellipse (cucul_canvas_t ∗ *cv*, int *xo*, int *yo*, int *a*, int *b*, char const ∗ *str*)

**Parameters:**

***cv*** The handle to the libcucul canvas.

***xo*** Center X coordinate.

***yo*** Center Y coordinate.

***a*** Ellipse X radius.

***b*** Ellipse Y radius.

***str*** UTF-8 string representing the character that should be used to draw the ellipse outline.

**Returns:**

void

#### 2.6.2.7 void cucul_draw_thin_ellipse (cucul_canvas_t ∗ *cv*, int *xo*, int *yo*, int *a*, int *b*)

**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:**

void

#### 2.6.2.8 void cucul_fill_ellipse (cucul_canvas_t ∗ *cv*, int *xo*, int *yo*, int *a*, int *b*, char const ∗ *str*)

**Parameters:**

***cv*** The handle to the libcucul canvas.

***xo*** Center X coordinate.

***yo*** Center Y coordinate.

***a*** Ellipse X radius.

***b*** Ellipse Y radius.

***str*** UTF-8 string representing the character that should be used to fill the ellipse.

**Returns:**

void

#### 2.6.2.9 void cucul_draw_box (cucul_canvas_t ∗ *cv*, int *x1*, int *y1*, int *x2*, int *y2*, char const ∗ *str*)

**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.

*str* UTF-8 string containing the character to use to draw the box.

**Returns:**

void

#### 2.6.2.10 void cucul_draw_thin_box (cucul_canvas_t ∗ *cv*, int *x1*, int *y1*, int *x2*, int *y2*)

**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:**

void

#### 2.6.2.11 void cucul_fill_box (cucul_canvas_t ∗ *cv*, int *x1*, int *y1*, int *x2*, int *y2*, char const ∗ *str*)

**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.

*str* UTF-8 string containing the character to fill the box with.

**Returns:**

void

#### 2.6.2.12 void cucul_draw_triangle (cucul_canvas_t ∗ *cv*, int *x1*, int *y1*, int *x2*, int *y2*, int *x3*, int *y3*, char const ∗ *str*)

**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.

*str* UTF-8 string representing the character that should be used to draw the triangle outline.

**Returns:**

void

#### 2.6.2.13 void cucul_draw_thin_triangle (cucul_canvas_t ∗ *cv*, int *x1*, int *y1*, int *x2*, int *y2*, int *x3*, int *y3*)

**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:**

void

#### 2.6.2.14 void cucul_fill_triangle (cucul_canvas_t ∗ *cv*, int *x1*, int *y1*, int *x2*, int *y2*, int *x3*, int *y3*, char const ∗ *str*)

**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.

*str* UTF-8 string representing the character that should be used to fill the triangle.

**Returns:**

void

## 2.7 libcucul canvas frame handling

### Functions

- unsigned int cucul_get_canvas_frame_count (cucul_canvas_t ∗)

  *Get the number of frames in a canvas.*

- void cucul_set_canvas_frame (cucul_canvas_t ∗, unsigned int)

  *Activate a given canvas frame.*

- void cucul_create_canvas_frame (cucul_canvas_t ∗, unsigned int)

  *Add a frame to a canvas.*

- void cucul_free_canvas_frame (cucul_canvas_t ∗, unsigned int)

  *Remove a frame from a canvas.*

### 2.7.1 Detailed Description

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

### 2.7.2 Function Documentation

#### 2.7.2.1 unsigned int cucul_get_canvas_frame_count (cucul_canvas_t ∗ *cv*)

This function returns the current canvas frame count.

**Parameters:**

***cv*** A libcucul canvas

**Returns:**

The frame count

#### 2.7.2.2 void cucul_set_canvas_frame (cucul_canvas_t ∗ *cv*, unsigned int *frame*)

This function sets the active canvas frame. All subsequent drawing operations will be performed on that frame. The current painting context set by cucul_set_color() or cucul_set_truecolor() is inherited.

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

**Parameters:**

***cv*** A libcucul canvas

***frame*** The canvas frame to activate

#### 2.7.2.3 void cucul_create_canvas_frame (cucul_canvas_t ∗ *cv*, unsigned int *frame*)

This function creates a new frame within the given canvas. Its contents 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 the 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.

**Parameters:**

*cv* A libcucul canvas

*frame* The index where to insert the new frame

#### 2.7.2.4 void cucul_free_canvas_frame (cucul_canvas_t ∗ *cv*, unsigned int *frame*)

This function deletes a frame from a given canvas.

It is not legal to remove the last frame from a canvas. Such a request will be ignored by cucul_free_-canvas_frame().

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.

**Parameters:**

*cv* A libcucul canvas

*frame* The index of the frame to delete

## 2.8 libcucul bitmap dithering

### Functions

- cucul_dither_t ∗ cucul_create_dither (unsigned int, unsigned int, unsigned int, unsigned int, unsigned int, unsigned int, unsigned int, unsigned int)

  *Create an internal dither object.*

- void 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.*

- void cucul_set_dither_brightness (cucul_dither_t ∗, float)

  *Set the brightness of a dither object.*

- void cucul_set_dither_gamma (cucul_dither_t ∗, float)

  *Set the gamma of a dither object.*

- void cucul_set_dither_contrast (cucul_dither_t ∗, float)

  *Set the contrast of a dither object.*

- void cucul_set_dither_invert (cucul_dither_t ∗, int)

  *Invert colors of dither.*

- void 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.*

- void 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.*

- void 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.*

- void 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.*

- void cucul_dither_bitmap (cucul_canvas_t ∗, int, int, int, int, cucul_dither_t const ∗, void ∗)

  *Dither a bitmap on the canvas.*

- void cucul_free_dither (cucul_dither_t ∗)

  *Free the memory associated with a dither.*

### 2.8.1 Detailed Description

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

### 2.8.2 Function Documentation

#### 2.8.2.1 cucul_dither_t∗ cucul_create_dither (unsigned int *bpp*, unsigned int *w*, unsigned int *h*, unsigned int *pitch*, unsigned int *rmask*, unsigned int *gmask*, unsigned int *bmask*, unsigned 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.

**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, or NULL upon error.

#### 2.8.2.2 void 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).

**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.

#### 2.8.2.3 void cucul_set_dither_brightness (cucul_dither_t ∗ *d*, float *brightness*)

Set the brightness of dither.

**Parameters:**

***d*** Dither object.

***brightness*** brightness value.

#### 2.8.2.4 void cucul_set_dither_gamma (cucul_dither_t ∗ *d*, float *gamma*)

Set the gamma of dither.

**Parameters:**

***d*** Dither object.

***gamma*** Gamma value.

#### 2.8.2.5 void cucul_set_dither_contrast (cucul_dither_t ∗ *d*, float *contrast*)

Set the contrast of dither.

**Parameters:**

***d*** Dither object.

***contrast*** contrast value.

#### 2.8.2.6 void cucul_set_dither_invert (cucul_dither_t ∗ *d*, int *value*)

Invert colors of dither

**Parameters:**

***d*** Dither object.

***value*** 0 for normal behaviour, 1 for invert

#### 2.8.2.7 void 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"`: simple prefilter antialiasing. This is the default value.

**Parameters:**

*d* Dither object.

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

#### 2.8.2.8 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.

**Parameters:**

*d* Dither object.

**Returns:**

An array of strings.

#### 2.8.2.9 void 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"`: use the 16 ANSI colours for both the characters and the background. This is the default value.

**Parameters:**

*d* Dither object.

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

#### 2.8.2.10 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.

**Parameters:**

*d* Dither object.

**Returns:**

An array of strings.

#### 2.8.2.11 void 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"`: 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.

**Parameters:**

*d* Dither object.

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

#### 2.8.2.12 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.

**Parameters:**

*d* Dither object.

**Returns:**

An array of strings.

#### 2.8.2.13 void 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.

**Parameters:**

***d*** Dither object.

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

#### 2.8.2.14 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.

**Parameters:**

***d*** Dither object.

**Returns:**

An array of strings.

#### 2.8.2.15 void 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.

**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.

#### 2.8.2.16 void cucul_free_dither (cucul_dither_t ∗ *d*)

Free the memory allocated by cucul_create_dither().

**Parameters:**

***d*** Dither object.

## 2.9 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 maximum glyph width.*

- unsigned int cucul_get_font_height (cucul_font_t ∗)

  *Get a font's maximum glyph height.*

- void cucul_render_canvas (cucul_canvas_t ∗, cucul_font_t ∗, void ∗, unsigned int, unsigned int, unsigned int)

  *Render the canvas onto an image buffer.*

- void cucul_free_font (cucul_font_t ∗)

  *Free a font structure.*

### 2.9.1 Detailed Description

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

### 2.9.2 Function Documentation

#### 2.9.2.1 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().

**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.

#### 2.9.2.2 char const∗ const∗ cucul_get_font_list (void)

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

**Returns:**

An array of strings.

#### 2.9.2.3 unsigned int cucul_get_font_width (cucul_font_t ∗ *f*)

This function returns the maximum value for the current font's glyphs

**Parameters:**

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

**Returns:**

The maximum glyph width.

#### 2.9.2.4 unsigned int cucul_get_font_height (cucul_font_t ∗ *f*)

This function returns the maximum value for the current font's glyphs

**Parameters:**

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

**Returns:**

The maximum glyph height.

#### 2.9.2.5 void 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.

**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.

#### 2.9.2.6 void 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.

**Parameters:**

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

## 2.10 libcucul importers/exporters from/to various formats

### Functions

- cucul_buffer_t ∗ cucul_export_canvas (cucul_canvas_t ∗, char const ∗)

  *Export a canvas into a foreign format.*

- char const ∗const ∗ cucul_get_export_list (void)

  *Get available export formats.*

- cucul_canvas_t ∗ cucul_import_canvas (void const ∗, unsigned int, char const ∗)

  *Import a buffer into a canvas.*

- char const ∗const ∗ cucul_get_import_list (void)

  *Get available import formats.*

### 2.10.1 Detailed Description

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

### 2.10.2 Function Documentation

#### 2.10.2.1 cucul_buffer_t∗ cucul_export_canvas (cucul_canvas_t ∗ *cv*, char const ∗ *format*)

This function exports a libcucul canvas into various foreign formats such as ANSI art, HTML, IRC colours, etc. One should use cucul_get_buffer_data() and cucul_get_buffer_size() to access the buffer contents. The allocated data is valid until cucul_free_buffer() is called.

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.

**Parameters:**

***cv*** A libcucul canvas

***format*** A string describing the requested output format.

#### 2.10.2.2 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_canvas(), and a string containing the natural language description for that export format.

**Returns:**

An array of strings.

#### 2.10.2.3 cucul_canvas_t∗ cucul_import_canvas (void const ∗ *data*, unsigned int *size*, char const ∗ *format*)

This function imports a memory area into an internal libcucul canvas.

Valid values for `format` are:

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

**Parameters:**

***data*** The memory area to be loaded into a canvas.

***size*** The length of the memory area.

***format*** A string describing the input format.

**Returns:**

A libcucul canvas, or NULL in case of error.

#### 2.10.2.4 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.

**Returns:**

An array of strings.

## 2.11 libcaca basic functions

### Functions

- caca_display_t ∗ caca_create_display (cucul_canvas_t ∗)
  *Attach a caca graphical context to a cucul canvas.*
- void caca_free_display (caca_display_t ∗)
  *Detach a caca graphical context from a cucul backend context.*
- void caca_set_delay (caca_display_t ∗, unsigned int)
  *Set the refresh delay.*
- void caca_refresh_display (caca_display_t ∗)
  *Flush pending changes and redraw the screen.*
- unsigned int caca_get_rendertime (caca_display_t ∗)
  *Get the 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.*

### 2.11.1 Detailed Description

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

### 2.11.2 Function Documentation

#### 2.11.2.1 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.

**Parameters:**
***cv*** The cucul cavas.

**Returns:**
The caca graphical context or NULL if an error occurred.

#### 2.11.2.2 void 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.

**Parameters:**

*dp* The libcaca graphical context.

#### 2.11.2.3 void caca_set_delay (caca_display_t * *dp*, unsigned int *usec*)

This function sets 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.

**Parameters:**

*dp* The libcaca graphical context.

*usec* The refresh delay in microseconds.

#### 2.11.2.4 void caca_refresh_display (caca_display_t * *dp*)

This function flushes all graphical operations and prints them to the screen. Nothing will show on the screen until caca_refresh_display() is called.

If caca_set_delay() 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_delay(), the second call will be delayed before performing the screen refresh.

**Parameters:**

*dp* The libcaca graphical context.

#### 2.11.2.5 unsigned int caca_get_rendertime (caca_display_t * *dp*)

This function returns 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_delay(), the average rendering time will not be considerably shorter than the requested delay even if the real rendering time was shorter.

**Parameters:**

*dp* The libcaca graphical context.

**Returns:**

The render time in microseconds.

#### 2.11.2.6 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.

**Parameters:**

*dp* The libcaca graphical context.

**Returns:**
The display width.

#### 2.11.2.7 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.

**Parameters:**
***dp*** The libcaca graphical context.

**Returns:**
The display height.

#### 2.11.2.8 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 X11 and Win32 drivers.

**Parameters:**
***dp*** The libcaca graphical context.
***title*** The desired display title.

**Returns:**
0 upon success, a non-zero value if an error occurs.

## 2.12 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.*
- void caca_set_mouse (caca_display_t ∗, int)
  *Show or hide the mouse pointer.*

### 2.12.1 Detailed Description

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

### 2.12.2 Function Documentation

#### 2.12.2.1 int caca_get_event (caca_display_t ∗ *dp*, unsigned int *event_mask*, caca_event_t ∗ *ev*, int *timeout*)

This function polls the event queue for mouse or keyboard events matching the event mask and returns the first matching event. Non-matching events are discarded. `event_mask` must have a non-zero value.

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.

**Parameters:**

*dp* The libcaca graphical context.

*event_mask* Bitmask of requested events.

*timeout* A timeout value in microseconds

*ev* A pointer to a caca_event structure, or NULL.

**Returns:**

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

#### 2.12.2.2 unsigned int caca_get_mouse_x (caca_display_t ∗ *dp*)

This function returns 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.

**Parameters:**

*dp* The libcaca graphical context.

**Returns:**

The X mouse coordinate.

#### 2.12.2.3 unsigned int caca_get_mouse_y (caca_display_t ∗ *dp*)

This function returns 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.

**Parameters:**

*dp* The libcaca graphical context.

**Returns:**

The Y mouse coordinate.

#### 2.12.2.4 void caca_set_mouse (caca_display_t ∗ *dp*, int *flag*)

This function shows or hides the mouse pointer, for devices that support it.

**Parameters:**

*dp* The libcaca graphical context.

*flag* 0 hides the pointer, 1 shows the system's default pointer (usually an arrow). Other values are reserved for future use.

# 3 libcaca Data Structure Documentation

## 3.1 caca_event Struct Reference

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 }

### Data Fields

- enum caca_event::caca_event_type **type**
- 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 ucs4
    char utf8 [8]
  } key
} data
```

### 3.1.1 Detailed Description

This structure is filled by caca_get_event() when an event is received. The *type* field is always valid. The validity of the *data* union depends on the value of the *type* field:

- **CACA_EVENT_NONE:** no other 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.ucs4* and *data.key.utf8* fields are also filled and contain respectively the UCS-4/UTF-32 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.

### 3.1.2 Member Enumeration Documentation

#### 3.1.2.1 enum caca_event::caca_event_type

**Enumerator:**

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

# 4 libcaca File Documentation

## 4.1 caca.h File Reference

The *libcaca* public header.

### Data Structures

- struct caca_event

  *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_BACKSPACE = 0x08, CACA_KEY_TAB = 0x09, CACA_KEY_RETURN = 0x0d,

  CACA_KEY_PAUSE = 0x13, 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.*

- void caca_free_display (caca_display_t ∗)

  *Detach a caca graphical context from a cucul backend context.*

- void caca_set_delay (caca_display_t ∗, unsigned int)

  *Set the refresh delay.*

- void caca_refresh_display (caca_display_t ∗)

  *Flush pending changes and redraw the screen.*

- unsigned int caca_get_rendertime (caca_display_t ∗)

  *Get the 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.*

- void caca_set_mouse (caca_display_t ∗, int)

  *Show or hide the mouse pointer.*

### 4.1.1 Detailed Description

**Version:**

$Id: caca.h 613 2006-04-21 18:03:22Z 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* context

#### 4.1.3.2 typedef struct caca_event caca_event_t

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 ASCII equivalent.

**Enumerator:**

| | |
|---|---|
| ***CACA_KEY_UNKNOWN*** | Unknown key. |
| ***CACA_KEY_BACKSPACE*** | The backspace key. |
| ***CACA_KEY_TAB*** | The tabulation key. |
| ***CACA_KEY_RETURN*** | The return key. |
| ***CACA_KEY_PAUSE*** | The pause 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_COLOR_BLACK 0x00
- #define CUCUL_COLOR_BLUE 0x01
- #define CUCUL_COLOR_GREEN 0x02
- #define CUCUL_COLOR_CYAN 0x03
- #define CUCUL_COLOR_RED 0x04
- #define CUCUL_COLOR_MAGENTA 0x05
- #define CUCUL_COLOR_BROWN 0x06
- #define CUCUL_COLOR_LIGHTGRAY 0x07
- #define CUCUL_COLOR_DARKGRAY 0x08
- #define CUCUL_COLOR_LIGHTBLUE 0x09
- #define CUCUL_COLOR_LIGHTGREEN 0x0a
- #define CUCUL_COLOR_LIGHTCYAN 0x0b
- #define CUCUL_COLOR_LIGHTRED 0x0c
- #define CUCUL_COLOR_LIGHTMAGENTA 0x0d
- #define CUCUL_COLOR_YELLOW 0x0e
- #define CUCUL_COLOR_WHITE 0x0f
- #define CUCUL_COLOR_DEFAULT 0x10
- #define CUCUL_COLOR_TRANSPARENT 0x20

### Typedefs

- typedef cucul_canvas cucul_canvas_t
- typedef cucul_dither cucul_dither_t
- typedef cucul_buffer cucul_buffer_t
- typedef cucul_font cucul_font_t

### Functions

- cucul_canvas_t ∗ cucul_create_canvas (unsigned int, unsigned int)

  *Initialise a* libcucul *canvas.*

- void 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.*

- void cucul_free_canvas (cucul_canvas_t ∗)

  *Uninitialise* libcucul.

- int cucul_rand (int, int)

  *Generate a random integer within a range.*

- unsigned long int cucul_get_buffer_size (cucul_buffer_t ∗)

*Get the buffer size.*

- void ∗ cucul_get_buffer_data (cucul_buffer_t ∗)

  *Get the buffer data.*

- void cucul_free_buffer (cucul_buffer_t ∗)

  *Free a buffer.*

- void cucul_set_color (cucul_canvas_t ∗, unsigned char, unsigned char)

  *Set the default colour pair.*

- void cucul_set_truecolor (cucul_canvas_t ∗, unsigned int, unsigned int)

  *Set the default colour pair (truecolor version).*

- char const ∗ cucul_get_color_name (unsigned int)

  *Translate a colour index into the colour's name.*

- void cucul_putchar (cucul_canvas_t ∗, int, int, char)

  *Print an ASCII character.*

- void cucul_putstr (cucul_canvas_t ∗, int, int, char const ∗)

  *Print a string.*

- void cucul_printf (cucul_canvas_t ∗, int, int, char const ∗,...)

  *Print a formated string.*

- void cucul_clear_canvas (cucul_canvas_t ∗)

  *Clear the canvas.*

- void cucul_blit (cucul_canvas_t ∗, int, int, cucul_canvas_t const ∗, cucul_canvas_t const ∗)

  *Blit a canvas onto another one.*

- void cucul_invert (cucul_canvas_t ∗)

  *Invert a canvas' colours.*

- void cucul_flip (cucul_canvas_t ∗)

  *Flip a canvas horizontally.*

- void cucul_flop (cucul_canvas_t ∗)

  *Flip a canvas vertically.*

- void cucul_rotate (cucul_canvas_t ∗)

  *Rotate a canvas.*

- void cucul_draw_line (cucul_canvas_t ∗, int, int, int, int, char const ∗)

  *Draw a line on the canvas using the given character.*

- void cucul_draw_polyline (cucul_canvas_t ∗, int const x[ ], int const y[ ], int, char const ∗)

  *Draw a polyline.*

- void cucul_draw_thin_line (cucul_canvas_t ∗, int, int, int, int)

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

- void cucul_draw_thin_polyline (cucul_canvas_t ∗, int const x[ ], int const y[ ], int)

  *Draw an ASCII art thin polyline.*

- void cucul_draw_circle (cucul_canvas_t ∗, int, int, int, char const ∗)

  *Draw a circle on the canvas using the given character.*

- void cucul_draw_ellipse (cucul_canvas_t ∗, int, int, int, int, char const ∗)

  *Draw an ellipse on the canvas using the given character.*

- void cucul_draw_thin_ellipse (cucul_canvas_t ∗, int, int, int, int)

  *Draw a thin ellipse on the canvas.*

- void cucul_fill_ellipse (cucul_canvas_t ∗, int, int, int, int, char const ∗)

  *Fill an ellipse on the canvas using the given character.*

- void cucul_draw_box (cucul_canvas_t ∗, int, int, int, int, char const ∗)

  *Draw a box on the canvas using the given character.*

- void cucul_draw_thin_box (cucul_canvas_t ∗, int, int, int, int)

  *Draw a thin box on the canvas.*

- void cucul_fill_box (cucul_canvas_t ∗, int, int, int, int, char const ∗)

  *Fill a box on the canvas using the given character.*

- void cucul_draw_triangle (cucul_canvas_t ∗, int, int, int, int, int, int, char const ∗)

  *Draw a triangle on the canvas using the given character.*

- void cucul_draw_thin_triangle (cucul_canvas_t ∗, int, int, int, int, int, int)

  *Draw a thin triangle on the canvas.*

- void cucul_fill_triangle (cucul_canvas_t ∗, int, int, int, int, int, int, char const ∗)

  *Fill a triangle on the canvas using the given character.*

- unsigned int cucul_get_canvas_frame_count (cucul_canvas_t ∗)

  *Get the number of frames in a canvas.*

- void cucul_set_canvas_frame (cucul_canvas_t ∗, unsigned int)

  *Activate a given canvas frame.*

- void cucul_create_canvas_frame (cucul_canvas_t ∗, unsigned int)

  *Add a frame to a canvas.*

- void cucul_free_canvas_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 int, unsigned int, unsigned int, unsigned int)

*Create an internal dither object.*

- void 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.*

- void cucul_set_dither_brightness (cucul_dither_t ∗, float)

  *Set the brightness of a dither object.*

- void cucul_set_dither_gamma (cucul_dither_t ∗, float)

  *Set the gamma of a dither object.*

- void cucul_set_dither_contrast (cucul_dither_t ∗, float)

  *Set the contrast of a dither object.*

- void cucul_set_dither_invert (cucul_dither_t ∗, int)

  *Invert colors of dither.*

- void 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.*

- void 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.*

- void 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.*

- void 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.*

- void cucul_dither_bitmap (cucul_canvas_t ∗, int, int, int, int, cucul_dither_t const ∗, void ∗)

  *Dither a bitmap on the canvas.*

- void 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 maximum glyph width.*

- unsigned int cucul_get_font_height (cucul_font_t ∗)

  *Get a font's maximum glyph height.*

- void cucul_render_canvas (cucul_canvas_t ∗, cucul_font_t ∗, void ∗, unsigned int, unsigned int, unsigned int)

  *Render the canvas onto an image buffer.*

- void cucul_free_font (cucul_font_t ∗)

  *Free a font structure.*

- cucul_buffer_t ∗ cucul_export_canvas (cucul_canvas_t ∗, char const ∗)

  *Export a canvas into a foreign format.*

- char const ∗const ∗ cucul_get_export_list (void)

  *Get available export formats.*

- cucul_canvas_t ∗ cucul_import_canvas (void const ∗, unsigned int, char const ∗)

  *Import a buffer into a canvas.*

- char const ∗const ∗ cucul_get_import_list (void)

  *Get available import formats.*

### 4.2.1 Detailed Description

**Version:**
$Id: cucul.h 631 2006-04-22 19:11:25Z 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* context

#### 4.2.3.2 typedef struct cucul_dither cucul_dither_t

dither structure

#### 4.2.3.3 typedef struct cucul_buffer cucul_buffer_t

data buffer structure

#### 4.2.3.4 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 and 0.99

- 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

### 5.2.2 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

### 5.2.3 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 moiré circles animation example
- MSVC project file

### 5.2.4 Changes between 0.6 and 0.7

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

### 5.2.5 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

### 5.2.6 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

### 5.2.7 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

### 5.2.8 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

### 5.2.9 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

### 5.2.10 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 multi-frame canvases in the caca exporter

#### 5.4.1.2 API-independent stuff

- support for transparency (CUCUL_COLOR_TRANSPARENT)
- 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).

- support for double width glyphs (also needs some libcaca changes)
- better mask support in cucul_blit()
- factor internal Unicode character conversions, especially UCS4 -> UTF-8
- optimise exporters so that they do not allocate huge blocks of memory when they only need half of it.
- ASCII/ANSI art loading functions (maybe load them as sprites)

### 5.4.2 libcaca

#### 5.4.2.1 API-dependent stuff

- text edit widget with cursor support (I'm unsure about this, it seems pretty difficult)
- rename caca_set_delay into something like caca_set_rendertime.

#### 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
- Unicode support for GL
- and Jylam wants a framebuffer output

### 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

- Does not exist yet, but I want it. A modern ANSI editor that can also do Unicode.

### 5.4.9 CUCUlet

- Does not exist yet, but I want it. A replacement for FIGlet that can also do Unicode
- Colour support, of course: outputs to IRC, ANSI, HTML...
- Can open FIGlet fonts

## 5.5 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.

### 5.5.1 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(cv, CUCUL_COLOR_BLACK,
                        CUCUL_COLOR_WHITE);
    /* Draw a string at (0, 0) */
    cucul_putstr(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_** .

### 5.5.2 Function equivalence list

#### 5.5.2.1 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()**: unchanged.
- **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()**: unchanged.
- **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.

#### 5.5.2.2 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.

#### 5.5.2.3 Character printing

- **caca_set_color()**: use cucul_set_color() or cucul_set_truecolor().
- **caca_get_fg_color()**: deprecated.
- **caca_get_bg_color()**: deprecated.
- **caca_get_color_name()**: use cucul_get_color_name().
- **caca_putchar()**: use cucul_putchar().
- **caca_putstr()**: use cucul_putstr().
- **caca_printf()**: use cucul_printf().
- **caca_clear()**: use cucul_clear_canvas().

**5.5.2.4 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().
- **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().

#### 5.5.2.5 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.

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

- **caca_load_sprite**(): use cucul_import_canvas().
- **caca_get_sprite_frames**(): use cucul_get_canvas_frame_count().
- **caca_get_sprite_width**(): use cucul_get_canvas_width().
- **caca_get_sprite_height**(): use cucul_get_canvas_height().
- **caca_get_sprite_dx**(): this function is now deprecated.
- **caca_get_sprite_dy**(): this function is now deprecated.
- **caca_draw_sprite**(): use cucul_set_canvas_frame() and cucul_blit().
- **caca_free_sprite**(): use cucul_free_canvas().

**5.5.2.7 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().

### 5.5.3 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 will be removed in the future.

## 5.6 Coding style

### 5.6.1 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

### 5.6.2 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;

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

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

### 5.6.3 C++ coding style

Nothing here yet.

## 5.7 A libcucul and libcaca tutorial

Super short example:

```
#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(cv, CUCUL_COLOR_BLACK, CUCUL_COLOR_WHITE);
    /* Draw a string at coordinates (0, 0) */
    cucul_putstr(cv, 0, 0, "This is a message");
    /* Refresh display */
    caca_refresh_display();
    /* 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;
}
```

## 5.8 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 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