libtcod Documentation

Release 1.16.7

Richard Tew

Apr 16, 2021

CORE

1 Colors 3

2 Console 9

3 System layer 31

4 Line Drawing Toolkit 33

5 Upgrading 37

6 Getting Started 39

Index 41

Note: These docs are incomplete. Several functions and features are better documented by the older 1.6.4 docs which you can find here.

CHAPTER

ONE

COLORS

Note: These docs are incomplete. Several functions and features are better documented by the older 1.6.4 docs which you can find here.

libtcod uses 32-bit color, therefore your OS desktop must also use 32-bit color. A color is defined by its red, green and blue component between 0 and 255.

You can use the following predefined colors (hover over a color to see its full name and R,G,B values):

INSERT COLOUR TABLE IN A PAINLESS MANNER

1.1 Create your own colors

You can create your own colours using a set of constructors, both for RGB and HSV values.

```
/* RGB */
TCOD_color_t my_color= { 24, 64, 255 };
TCOD_color_t my_other_color = TCOD_color_RGB(24, 64, 255);
/* HSV */
TCOD_color_t my_yet_another_color = TCOD_color_HSV(321.0f, 0.7f, 1.0f);
```

```
// RGB
TCODColor myColor(24, 64, 255);
// HSV
TCODColor myOtherColor(321.0f, 0.7f, 1.0f);
```

```
my_color = libtcod.Color(24, 64, 255)
```

1.2 Compare two colors

bool **TCOD_color_equals** (TCOD_color_t *c1*, TCOD_color_t *c2*)
Return a true value if c1 and c2 are equal.

1.3 Add and subtract Colors

TCOD_color_t **TCOD_color_add**(TCOD_color_t *c1*, TCOD_color_t *c2*)
Add two colors together and return the result.

Parameters

- **c1** – The first color.
- **c2** – The second color.

Returns A new TCOD_color_t struct with the result.

TCOD_color_t **TCOD_color_subtract**(TCOD_color_t *c1*, TCOD_color_t *c2*)
Subtract c2 from c1 and return the result.

Parameters

- **c1** – The first color.
- **c2** – The second color.

Returns A new TCOD_color_t struct with the result.

1.4 Multiply Colors together

TCOD_color_t **TCOD_color_multiply**(TCOD_color_t *c1*, TCOD_color_t *c2*)
Multiply two colors together and return the result.

Parameters

- **c1** – The first color.
- **c2** – The second color.

Returns A new TCOD_color_t struct with the result.

TCOD_color_t **TCOD_color_multiply_scalar**(TCOD_color_t *c1*, float *value*)
Multiply a color with a scalar value and return the result.

Parameters

- **c1** – The color to multiply.
- **value** – The scalar float.

Returns A new TCOD_color_t struct with the result.

1.5 Interpolate between two colors

TCOD_color_t **TCOD_color_lerp**(TCOD_color_t *c1*, TCOD_color_t *c2*, float *coef*)
Interpolate two colors together and return the result.

Parameters

- **c1** – The first color (where coef if 0)
- **c2** – The second color (where coef if 1)
- **coef** – The coefficient.

Returns A new TCOD_color_t struct with the result.

1.6 Define a color by its hue, saturation and value

After this function is called, the r,g,b fields of the color are calculated according to the h,s,v parameters.

void **TCOD_color_set_HSV**(TCOD_color_t *color, float *hue*, float *saturation*, float *value*)
Sets a colors values from HSV values.

Parameters

- **color** – The color to be changed.
- **hue** – The colors hue (in degrees.)
- **saturation** – The colors saturation (from 0 to 1)
- **value** – The colors value (from 0 to 1)

These functions set only a single component in the HSV color space.

void **TCOD_color_set_hue**(TCOD_color_t *color, float *hue*)
Change a colors hue.

Parameters

- **color** – Pointer to a color struct.
- **hue** – The hue in degrees.

void **TCOD_color_set_saturation**(TCOD_color_t *color, float *saturation*)
Change a colors saturation.

Parameters

- **color** – Pointer to a color struct.
- **saturation** – The desired saturation value.

void **TCOD_color_set_value**(TCOD_color_t *color, float *value*)
Change a colors value.

Parameters

- **color** – Pointer to a color struct.
- **value** – The desired value.

1.7 Get a color hue, saturation and value components

void **TCOD_color_get_HSV**(TCOD_color_t *color*, float **hue*, float **saturation*, float **value*)
Get a set of HSV values from a color.

The hue, saturation, and value parameters can not be NULL pointers,

Parameters

- **color** – The color
- **hue** – Pointer to a float, filled with the hue. (degrees)
- **saturation** – Pointer to a float, filled with the saturation. (0 to 1)

- **value** – Pointer to a float, filled with the value. (0 to 1)

Should you need to extract only one of the HSV components, these functions are what you should call. Note that if you need all three values, it's way less burdensome for the CPU to call *TCODColor::getHSV()*.

float **TCOD_color_get_hue** (TCOD_color_t *color*)
Return a colors hue.

Parameters **color** – A color struct.

Returns The colors hue. (degrees)

float **TCOD_color_get_saturation** (TCOD_color_t *color*)
Return a colors saturation.

Parameters **color** – A color struct.

Returns The colors saturation. (0 to 1)

float **TCOD_color_get_value** (TCOD_color_t *color*)
Get a colors value.

Parameters **color** – A color struct.

Returns The colors value. (0 to 1)

1.8 Shift a color's hue up or down

The hue shift value is the number of grades the color's hue will be shifted. The value can be negative for shift left, or positive for shift right. Resulting values H < 0 and H >= 360 are handled automatically.

void **TCOD_color_shift_hue** (TCOD_color_t **color*, float *shift*)
Shift a colors hue by an amount.

Parameters

- **color** – Pointer to a color struct.
- **hue_shift** – The distance to shift the hue, in degrees.

1.9 Scale a color's saturation and value

void **TCOD_color_scale_HSV** (TCOD_color_t **color*, float *saturation_coef*, float *value_coef*)
Scale a colors saturation and value.

Parameters

- **color** – Pointer to a color struct.
- **saturation_coef** – Multiplier for this colors saturation.
- **value_coef** – Multiplier for this colors value.

1.10 Generate a smooth color map

You can define a color map from an array of color keys. Colors will be interpolated between the keys. 0 -> black 4 -> red 8 -> white Result:

INSERT TABLE.

void **TCOD_color_gen_map** (TCOD_color_t **map*, int *nb_key*, **const** TCOD_color_t **key_color*, **const** int **key_index*)

Generate an interpolated gradient of colors.

```
TCOD_color_t[256] gradient;
TCOD_color_t[4] key_color = {TCOD_black, TCOD_dark_amber,
                             TCOD_cyan, TCOD_white};
int[4] key_index = {0, 64, 192, 255};
TCOD_color_gen_map(&gradient, 4, &key_color, &key_index);
```

Parameters

- **map** – Array to fill with the new gradient.
- **nb_key** – The array size of the key_color and key_index parameters.
- **key_color** – An array of colors to use, in order.
- **key_index** – An array mapping key_color items to the map array.

CHAPTER

TWO

CONSOLE

Note: These docs are incomplete. Several functions and features are better documented by the older 1.6.4 docs which you can find here.

2.1 Initializing the console

2.1.1 Creating the game window

enum TCOD_renderer_t
The available renderers.

Values:

enumerator TCOD_RENDERER_GLSL
Alias for TCOD_RENDERER_OPENGL2.

enumerator TCOD_RENDERER_OPENGL
An OpenGL 1.1 implementation.

Performs worse than TCOD_RENDERER_GLSL without many benefits.

enumerator TCOD_RENDERER_SDL
A software based renderer.

The font file is loaded into RAM instead of VRAM in this implementation.

enumerator TCOD_RENDERER_SDL2
A new SDL2 renderer.

Allows the window to be resized.

You may set `SDL_HINT_RENDER_SCALE_QUALITY` to determine the tileset upscaling filter. Either nearest or linear. The hint will only take effect if it's set before this renderer is created.

New in version 1.8.

enumerator TCOD_RENDERER_OPENGL2
A new OpenGL 2.0 core renderer.

Allows the window to be resized.

You may set `SDL_HINT_RENDER_SCALE_QUALITY` to determine the tileset upscaling filter. Either nearest or linear. The hint will take effect on the next frame.

New in version 1.9.

Changed in version 1.11: This renderer now uses OpenGL 2.0 instead of 2.1.

Changed in version 1.16: Now checks the *SDL_HINT_RENDER_SCALE_QUALITY* hint.

enumerator TCOD_NB_RENDERERS

TCOD_Error **TCOD_console_init_root**(int *w*, int *h*, **const** char **title*, bool *fullscreen*, *TCOD_renderer_t* *renderer*)

Initialize the libtcod graphical engine.

You may want to call TCOD_console_set_custom_font BEFORE calling this function. By default this function loads libtcod's `terminal.png` image from the working directory.

Afterwards TCOD_quit must be called before the program exits.

Returns 0 on success, or -1 on an error, you can check the error with TCOD_sys_get_error()

`renderer` and vsync settings can be overridden by the `TCOD_RENDERER` or `TCOD_VSYNC` environment variables.

Valid case-sensitive options for `TCOD_RENDERER` are:

- sdl
- opengl
- glsl
- sdl2
- opengl2

Valid options for `TCOD_VSYNC` are `0` or `1`.

Changed in version 1.12: Now returns -1 on error instead of crashing.

Changed in version 1.13: Added the *TCOD_RENDERER* and *TCOD_VSYNC* overrides.

Parameters

- **w** – The width in tiles.
- **h** – The height in tiles.
- **title** – The title for the window.
- **fullscreen** – Fullscreen option.
- **renderer** – Which renderer to use when rendering the console.

void **TCOD_quit**(void)
Shutdown libtcod.

This must be called before your program exits.

New in version 1.8.

2.1.2 Using a custom bitmap font

enum TCOD_font_flags_t
These font flags can be OR'd together into a bit-field and passed to TCOD_console_set_custom_font.

Values:

enumerator TCOD_FONT_LAYOUT_ASCII_INCOL
Tiles are arranged in column-major order.

0 3 6 1 4 7 2 5 8

enumerator TCOD_FONT_LAYOUT_ASCII_INROW
Tiles are arranged in row-major order.

0 1 2 3 4 5 6 7 8

enumerator TCOD_FONT_TYPE_GREYSCALE
Converts all tiles into a monochrome gradient.

enumerator TCOD_FONT_TYPE_GRAYSCALE

enumerator TCOD_FONT_LAYOUT_TCOD
A unique layout used by some of libtcod's fonts.

enumerator TCOD_FONT_LAYOUT_CP437
Decode a code page 437 tileset into Unicode code-points.

New in version 1.10.

TCOD_Error **TCOD_console_set_custom_font**(**const** char **fontFile*, int *flags*, int *nb_char_horiz*, int *nb_char_vertic*)
Set a font image to be loaded during initialization.

`fontFile` will be case-sensitive depending on the platform.

Returns 0 on success, or -1 on an error, you can check the error with TCOD_sys_get_error()

Changed in version 1.12: Now returns -1 on error instead of crashing.

Parameters

- **fontFile** – The path to a font image.
- **flags** – A TCOD_font_flags_t bit-field describing the font image contents.
- **nb_char_horiz** – The number of columns in the font image.
- **nb_char_vertic** – The number of rows in the font image.

2.1.3 Using custom characters mapping

void **TCOD_console_map_ascii_code_to_font**(int *asciiCode*, int *fontCharX*, int *fontCharY*)
Remap a character code to a tile.

X,Y parameters are the coordinate of the tile, not pixel-coordinates.

Parameters

- **asciiCode** – Character code to modify.
- **fontCharX** – X tile-coordinate, starting from the left at zero.
- **fontCharY** – Y tile-coordinate, starting from the top at zero.

void **TCOD_console_map_ascii_codes_to_font**(int *asciiCode*, int *nbCodes*, int *fontCharX*, int *fontCharY*)
Remap a series of character codes to a row of tiles.

This function always assigns tiles in row-major order, even if the TCOD_FONT_LAYOUT_ASCII_INCOL flag was set.

Parameters

- **asciiCode** – The starting character code.
- **nbCodes** – Number of character codes to assign.
- **fontCharX** – First X tile-coordinate, starting from the left at zero.
- **fontCharY** – First Y tile-coordinate, starting from the top at zero.

void **TCOD_console_map_string_to_font**(**const** char **s*, int *fontCharX*, int *fontCharY*)
Remap a string of character codes to a row of tiles.

This function always assigns tiles in row-major order, even if the TCOD_FONT_LAYOUT_ASCII_INCOL flag was set.

Parameters

- **s** – A null-terminated string.
- **fontCharX** – First X tile-coordinate, starting from the left at zero.
- **fontCharY** – First Y tile-coordinate, starting from the top at zero.

2.1.4 Fullscreen mode

void **TCOD_console_set_fullscreen**(bool *fullscreen*)
Set the display to be full-screen or windowed.

Parameters **fullscreen** – If true the display will go full-screen.

bool **TCOD_console_is_fullscreen**(void)
Return true if the display is full-screen.

2.1.5 Communicate with the window manager

bool **TCOD_console_is_active**(void)
Return true if the window has keyboard focus.

bool **TCOD_console_has_mouse_focus**(void)
Return true if the window has mouse focus.

bool **TCOD_console_is_window_closed**(void)
Return true if the window is closing.

void **TCOD_console_set_window_title**(**const** char *title)
Change the title string of the active window.

Parameters **title** – A utf8 string.

2.1.6 libtcod's Credits

void **TCOD_console_credits**(void)

void **TCOD_console_credits_reset**(void)

bool **TCOD_console_credits_render**(int *x*, int *y*, bool *alpha*)

2.2 Drawing on the root console

2.2.1 Basic printing functions

void **TCOD_console_set_default_foreground**(TCOD_Console *con, TCOD_color_t *col*)

void **TCOD_console_set_default_background**(TCOD_Console *con, TCOD_color_t *col*)

void **TCOD_console_set_background_flag**(TCOD_Console *con, *TCOD_bkgnd_flag_t flag*)
Set a consoles default background flag.

Parameters

- **con** – A console pointer.
- **flag** – One of `TCOD_bkgnd_flag_t`.

void **TCOD_console_clear**(TCOD_Console *con)
Clear a console to its default colors and the space character code.

void **TCOD_console_put_char**(TCOD_Console *con, int *x*, int *y*, int *c*, *TCOD_bkgnd_flag_t flag*)
Draw a character on a console using the default colors.

Parameters

- **con** – A console pointer.
- **x** – The X coordinate, the left-most position being 0.
- **y** – The Y coordinate, the top-most position being 0.
- **c** – The character code to place.
- **flag** – A TCOD_bkgnd_flag_t flag.

void **TCOD_console_put_char_ex**(TCOD_Console *con, int x, int y, int c, TCOD_color_t fore, TCOD_color_t back)

Draw a character on the console with the given colors.

Parameters

- **con** – A console pointer.
- **x** – The X coordinate, the left-most position being 0.
- **y** – The Y coordinate, the top-most position being 0.
- **c** – The character code to place.
- **fore** – The foreground color.
- **back** – The background color. This color will not be blended.

void **TCOD_console_set_char**(TCOD_Console *con, int x, int y, int c)

Change a character on a console tile, without changing its colors.

Parameters

- **con** – A console pointer.
- **x** – The X coordinate, the left-most position being 0.
- **y** – The Y coordinate, the top-most position being 0.
- **c** – The character code to set.

void **TCOD_console_set_char_foreground**(TCOD_Console *con, int x, int y, TCOD_color_t col)

Change the foreground color of a console tile.

Parameters

- **con** – A console pointer.
- **x** – The X coordinate, the left-most position being 0.
- **y** – The Y coordinate, the top-most position being 0.
- **col** – The foreground color to set.

void **TCOD_console_set_char_background**(TCOD_Console *con, int x, int y, TCOD_color_t col, *TCOD_bkgnd_flag_t* flag)

Blend a background color onto a console tile.

Parameters

- **con** – A console pointer.
- **x** – The X coordinate, the left-most position being 0.
- **y** – The Y coordinate, the top-most position being 0.
- **col** – The background color to blend.
- **flag** – The blend mode to use.

void **TCOD_console_rect**(TCOD_Console *con, int x, int y, int w, int h, bool clear, *TCOD_bkgnd_flag_t* flag)

Draw a rectangle onto a console.

Parameters

- **con** – A console pointer.
- **x** – The starting region, the left-most position being 0.

- **y** – The starting region, the top-most position being 0.
- **rw** – The width of the rectangle.
- **rh** – The height of the rectangle.
- **clear** – If true the drawing region will be filled with spaces.
- **flag** – The blending flag to use.

void **TCOD_console_hline**(TCOD_Console *con, int x, int y, int l, TCOD_bkgnd_flag_t flag)
Draw a horizontal line using the default colors.

This function makes assumptions about the fonts character encoding. It will fail if the font encoding is not `cp437`.

Parameters

- **con** – A console pointer.
- **x** – The starting X coordinate, the left-most position being 0.
- **y** – The starting Y coordinate, the top-most position being 0.
- **l** – The width of the line.
- **flag** – The blending flag.

void **TCOD_console_vline**(TCOD_Console *con, int x, int y, int l, TCOD_bkgnd_flag_t flag)
Draw a vertical line using the default colors.

This function makes assumptions about the fonts character encoding. It will fail if the font encoding is not `cp437`.

Parameters

- **con** – A console pointer.
- **x** – The starting X coordinate, the left-most position being 0.
- **y** – The starting Y coordinate, the top-most position being 0.
- **l** – The height of the line.
- **flag** – The blending flag.

void **TCOD_console_print_frame**(TCOD_console_t *con*, int *x*, int *y*, int *w*, int *h*, bool *empty*, TCOD_bkgnd_flag_t *flag*, **const** char *fmt, ...)

2.2.2 Background effect flags

enum TCOD_bkgnd_flag_t
Background color blend modes.

Values:

enumerator TCOD_BKGND_NONE

enumerator TCOD_BKGND_SET

enumerator TCOD_BKGND_MULTIPLY

enumerator TCOD_BKGND_LIGHTEN

enumerator TCOD_BKGND_DARKEN

enumerator TCOD_BKGND_SCREEN

enumerator TCOD_BKGND_COLOR_DODGE

enumerator TCOD_BKGND_COLOR_BURN

enumerator TCOD_BKGND_ADD

enumerator TCOD_BKGND_ADDA

enumerator TCOD_BKGND_BURN

enumerator TCOD_BKGND_OVERLAY

enumerator TCOD_BKGND_ALPH

enumerator TCOD_BKGND_DEFAULT

2.2.3 String printing alignment

enum TCOD_alignment_t
Print justification options.

Values:

enumerator TCOD_LEFT

enumerator TCOD_RIGHT

enumerator TCOD_CENTER

void **TCOD_console_set_alignment**(TCOD_Console *con, *TCOD_alignment_t* alignment)
Set a consoles default alignment.

Parameters

- **con** – A console pointer.
- **alignment** – One of TCOD_alignment_t

TCOD_alignment_t **TCOD_console_get_alignment**(TCOD_Console *con)
Return a consoles default alignment.

2.2.4 Printing functions using 8-bit encodings

void **TCOD_console_print**(TCOD_Console *con, int x, int y, **const** char *fmt, ...)
Print a string on a console, using default colors and alignment.

Parameters

- **con** – A console pointer.
- **x** – The starting X coordinate, the left-most position being 0.
- **y** – The starting Y coordinate, the top-most position being 0.
- **fmt** – A format string as if passed to printf.
- **...** – Variadic arguments as if passed to printf.

void **TCOD_console_print_ex**(TCOD_Console *con, int x, int y, *TCOD_bkgnd_flag_t* flag, *TCOD_alignment_t* alignment, **const** char *fmt, ...)
Print a string on a console, using default colors.

Parameters

- **con** – A console pointer.
- **x** – The starting X coordinate, the left-most position being 0.
- **y** – The starting Y coordinate, the top-most position being 0.
- **flag** – The blending flag.
- **alignment** – The font alignment to use.
- **fmt** – A format string as if passed to printf.
- **...** – Variadic arguments as if passed to printf.

int **TCOD_console_print_rect** (TCOD_Console *con, int x, int y, int w, int h, **const** char *fmt, ...)

Print a string on a console constrained to a rectangle, using default colors and alignment.

Parameters

- **con** – A console pointer.
- **x** – The starting X coordinate, the left-most position being 0.
- **y** – The starting Y coordinate, the top-most position being 0.
- **w** – The width of the region. If 0 then the maximum width will be used.
- **h** – The height of the region. If 0 then the maximum height will be used.
- **fmt** – A format string as if passed to printf.
- **...** – Variadic arguments as if passed to printf.

Returns The number of lines actually printed.

int **TCOD_console_print_rect_ex** (TCOD_Console *con, int x, int y, int w, int h, *TCOD_bkgnd_flag_t flag*, *TCOD_alignment_t alignment*, **const** char *fmt, ...)

Print a string on a console constrained to a rectangle, using default colors.

Parameters

- **con** – A console pointer.
- **x** – The starting X coordinate, the left-most position being 0.
- **y** – The starting Y coordinate, the top-most position being 0.
- **w** – The width of the region. If 0 then the maximum width will be used.
- **h** – The height of the region. If 0 then the maximum height will be used.
- **flag** – The blending flag.
- **alignment** – The font alignment to use.
- **fmt** – A format string as if passed to printf.
- **...** – Variadic arguments as if passed to printf.

Returns The number of lines actually printed.

int **TCOD_console_get_height_rect** (TCOD_Console *con, int x, int y, int w, int h, **const** char *fmt, ...)

Return the number of lines that would be printed by the.

Parameters

- **con** – A console pointer.

- **x** – The starting X coordinate, the left-most position being 0.
- **y** – The starting Y coordinate, the top-most position being 0.
- **w** – The width of the region. If 0 then the maximum width will be used.
- **h** – The height of the region. If 0 then the maximum height will be used.
- **fmt** – A format string as if passed to printf.
- **...** – Variadic arguments as if passed to printf.

Returns The number of lines that would have been printed.

2.2.5 Printing functions using UTF-8

TCOD_Error **TCOD_console_printf**(TCOD_Console *con, int x, int y, **const** char *fmt, ...)
Format and print a UTF-8 string to a console.

New in version 1.8.

Changed in version 1.16: Now returns a negative error code on failure.

TCOD_Error **TCOD_console_printf_ex**(TCOD_Console *con, int x, int y, TCOD_bkgnd_flag_t flag, TCOD_alignment_t alignment, **const** char *fmt, ...)
Format and print a UTF-8 string to a console.

New in version 1.8.

Changed in version 1.16: Now returns a negative error code on failure.

int **TCOD_console_printf_rect**(TCOD_Console *con, int x, int y, int w, int h, **const** char *fmt, ...)
Format and print a UTF-8 string to a console.

New in version 1.8.

Changed in version 1.16: Now returns a negative error code on failure.

int **TCOD_console_printf_rect_ex**(TCOD_Console *con, int x, int y, int w, int h, TCOD_bkgnd_flag_t flag, TCOD_alignment_t alignment, **const** char *fmt, ...)
Format and print a UTF-8 string to a console.

New in version 1.8.

Changed in version 1.16: Now returns a negative error code on failure.

int **TCOD_console_get_height_rect_fmt**(**struct** TCOD_Console *con, int x, int y, int w, int h, **const** char *fmt, ...)
Return the number of lines that would be printed by this formatted string.

New in version 1.8.

Changed in version 1.16: Now returns a negative error code on failure.

TCOD_Error **TCOD_console_printf_frame**(**struct** TCOD_Console *con, int x, int y, int w, int h, int empty, TCOD_bkgnd_flag_t flag, **const** char *fmt, ...)

Print a framed and optionally titled region to a console, using default colors and alignment.

This function uses Unicode box-drawing characters and a UTF-8 formatted string.

New in version 1.8.

Changed in version 1.16: Now returns a negative error code on failure.

2.2.6 Printing functions using wchar_t

Note: These functions say they are UTF, however they will behave as UCS2 or UCS4 depending on the platform.

void **TCOD_console_print_utf**(TCOD_Console *con, int x, int y, **const** wchar_t *fmt, ...)

Deprecated since version 1.8: Use *TCOD_console_printf()* instead.

void **TCOD_console_print_ex_utf**(TCOD_Console *con, int x, int y, TCOD_bkgnd_flag_t flag, TCOD_alignment_t alignment, **const** wchar_t *fmt, ...)

Deprecated since version 1.8: Use *TCOD_console_printf_ex()* instead.

int **TCOD_console_print_rect_utf**(TCOD_Console *con, int x, int y, int w, int h, **const** wchar_t *fmt, ...)

int **TCOD_console_print_rect_ex_utf**(TCOD_Console *con, int x, int y, int w, int h, TCOD_bkgnd_flag_t flag, TCOD_alignment_t alignment, **const** wchar_t *fmt, ...)

Deprecated since version 1.8: Use *TCOD_console_printf_rect_ex()* instead.

int **TCOD_console_get_height_rect_utf**(TCOD_Console *con, int x, int y, int w, int h, **const** wchar_t *fmt, ...)

Deprecated since version 1.8.

2.2.7 Reading the content of the console

int **TCOD_console_get_width**(**const** TCOD_Console *con)
Return the width of a console.

int **TCOD_console_get_height**(**const** TCOD_Console *con)
Return the height of a console.

int **TCOD_console_get_char**(**const** TCOD_Console *con, int x, int y)
Return a character code of a console at x,y.

Parameters

- **con** – A console pointer.
- **x** – The X coordinate, the left-most position being 0.
- **y** – The Y coordinate, the top-most position being 0.

Returns The character code.

TCOD_color_t **TCOD_console_get_char_foreground**(**const** TCOD_Console *con, int x, int y)
Return the foreground color of a console at x,y.

Parameters

- **con** – A console pointer.
- **x** – The X coordinate, the left-most position being 0.
- **y** – The Y coordinate, the top-most position being 0.

Returns A TCOD_color_t struct with a copy of the foreground color.

TCOD_color_t **TCOD_console_get_char_background**(**const** TCOD_Console *con, int x, int y)
Return the background color of a console at x,y.

Parameters

- **con** – A console pointer.
- **x** – The X coordinate, the left-most position being 0.
- **y** – The Y coordinate, the top-most position being 0.

Returns A TCOD_color_t struct with a copy of the background color.

TCOD_color_t **TCOD_console_get_default_foreground**(TCOD_Console *con)

TCOD_color_t **TCOD_console_get_default_background**(TCOD_Console *con)

TCOD_bkgnd_flag_t **TCOD_console_get_background_flag**(TCOD_Console *con)
Return a consoles default background flag.

2.2.8 Screen fading functions

void **TCOD_console_set_fade**(uint8_t *val*, TCOD_color_t *fade*)
Fade the color of the display.

Parameters

- **val** – Where at 255 colors are normal and at 0 colors are completely faded.
- **fadecol** – Color to fade towards.

uint8_t **TCOD_console_get_fade**(void)
Return the fade value.

Returns At 255 colors are normal and at 0 colors are completely faded.

TCOD_color_t **TCOD_console_get_fading_color**(void)
Return the fade color.

Returns The current fading color.

2.2.9 ASCII constants

enum TCOD_chars_t

Values:

enumerator TCOD_CHAR_HLINE

enumerator TCOD_CHAR_VLINE

enumerator TCOD_CHAR_NE

enumerator TCOD_CHAR_NW

enumerator TCOD_CHAR_SE

enumerator TCOD_CHAR_SW

enumerator TCOD_CHAR_TEEW

enumerator TCOD_CHAR_TEEE

enumerator TCOD_CHAR_TEEN

enumerator TCOD_CHAR_TEES

enumerator TCOD_CHAR_CROSS

enumerator TCOD_CHAR_DHLINE

enumerator TCOD_CHAR_DVLINE

enumerator TCOD_CHAR_DNE

enumerator TCOD_CHAR_DNW

enumerator TCOD_CHAR_DSE

enumerator TCOD_CHAR_DSW

enumerator TCOD_CHAR_DTEEW

enumerator TCOD_CHAR_DTEEE

enumerator TCOD_CHAR_DTEEN

enumerator TCOD_CHAR_DTEES

enumerator TCOD_CHAR_DCROSS

enumerator TCOD_CHAR_BLOCK1

enumerator TCOD_CHAR_BLOCK2

enumerator TCOD_CHAR_BLOCK3

enumerator TCOD_CHAR_ARROW_N

enumerator TCOD_CHAR_ARROW_S

enumerator TCOD_CHAR_ARROW_E

enumerator TCOD_CHAR_ARROW_W

enumerator TCOD_CHAR_ARROW2_N

enumerator TCOD_CHAR_ARROW2_S

enumerator TCOD_CHAR_ARROW2_E

enumerator TCOD_CHAR_ARROW2_W

enumerator TCOD_CHAR_DARROW_H

enumerator TCOD_CHAR_DARROW_V

enumerator TCOD_CHAR_CHECKBOX_UNSET

enumerator TCOD_CHAR_CHECKBOX_SET

enumerator TCOD_CHAR_RADIO_UNSET

enumerator TCOD_CHAR_RADIO_SET

enumerator TCOD_CHAR_SUBP_NW

enumerator TCOD_CHAR_SUBP_NE

enumerator TCOD_CHAR_SUBP_N

enumerator TCOD_CHAR_SUBP_SE

enumerator TCOD_CHAR_SUBP_DIAG

enumerator TCOD_CHAR_SUBP_E

enumerator TCOD_CHAR_SUBP_SW

enumerator TCOD_CHAR_SMILIE

enumerator TCOD_CHAR_SMILIE_INV

enumerator TCOD_CHAR_HEART

enumerator TCOD_CHAR_DIAMOND

enumerator TCOD_CHAR_CLUB

enumerator TCOD_CHAR_SPADE

enumerator TCOD_CHAR_BULLET

enumerator TCOD_CHAR_BULLET_INV

enumerator TCOD_CHAR_MALE

enumerator TCOD_CHAR_FEMALE

enumerator TCOD_CHAR_NOTE

enumerator TCOD_CHAR_NOTE_DOUBLE

enumerator TCOD_CHAR_LIGHT

enumerator TCOD_CHAR_EXCLAM_DOUBLE

enumerator TCOD_CHAR_PILCROW

enumerator TCOD_CHAR_SECTION

enumerator TCOD_CHAR_POUND

enumerator TCOD_CHAR_MULTIPLICATION

enumerator TCOD_CHAR_FUNCTION

enumerator TCOD_CHAR_RESERVED

enumerator TCOD_CHAR_HALF

enumerator TCOD_CHAR_ONE_QUARTER

enumerator TCOD_CHAR_COPYRIGHT

enumerator TCOD_CHAR_CENT

enumerator TCOD_CHAR_YEN

enumerator TCOD_CHAR_CURRENCY

enumerator TCOD_CHAR_THREE_QUARTERS

enumerator TCOD_CHAR_DIVISION

enumerator TCOD_CHAR_GRADE

enumerator TCOD_CHAR_UMLAUT

enumerator TCOD_CHAR_POW1

enumerator TCOD_CHAR_POW3

enumerator TCOD_CHAR_POW2

enumerator TCOD_CHAR_BULLET_SQUARE

2.3 Flushing the root console

TCOD_Error **TCOD_console_flush**(void)
Render and present the root console to the active display.

int **TCOD_sys_accumulate_console**(**const** TCOD_Console **console*)
Render a console over the display.

console can be any size, the active render will try to scale it to fit the screen.

The function will only work for the SDL2/OPENGL2 renderers.

Unlike *TCOD_console_flush()* this will not present the display. You will need to do that manually, likely with the SDL API.

Returns 0 on success, or a negative number on a failure such as the incorrect renderer being active.

New in version 1.11.

See also:

TCOD_sys_get_sdl_window() *TCOD_sys_get_sdl_renderer()*

2.4 Handling user input

2.4.1 Blocking user input

TCOD_key_t **TCOD_console_wait_for_keypress**(bool *flush*)
Wait for a key press event, then return it.

Do not solve input lag issues by arbitrarily dropping events!

Parameters **flush** – If 1 then the event queue will be cleared before waiting for the next event. This should always be 0.

Returns A *TCOD_key_t* struct with the most recent key data.

TCOD_event_t **TCOD_sys_wait_for_event**(int *eventMask*, *TCOD_key_t* **key*, *TCOD_mouse_t* **mouse*, bool *flush*)

Wait for a specific type of event.

This function also returns when the SDL window is being closed.

Parameters

- **eventMask** – A bit-mask of TCOD_event_t flags.
- **key** – Optional pointer to a *TCOD_key_t* struct.
- **mouse** – Optional pointer to a *TCOD_mouse_t* struct.
- **flush** – This should always be false.

Returns A TCOD_event_t flag showing which event was actually processed.

2.4.2 Non blocking user input

TCOD_key_t **TCOD_console_check_for_keypress**(int *flags*)

Return immediately with a recently pressed key.

Parameters **flags** – A TCOD_event_t bit-field, for example: `TCOD_EVENT_KEY_PRESS`

Returns A *TCOD_key_t* struct with a recently pressed key. If no event exists then the `vk` attribute will be `TCODK_NONE`

bool **TCOD_console_is_key_pressed**(*TCOD_keycode_t key*)

Return True if the libtcod keycode is held.

Deprecated since version 1.16: You should instead use SDL_GetKeyboardState to check if keys are held.

TCOD_event_t **TCOD_sys_check_for_event**(int *eventMask*, *TCOD_key_t* **key*, *TCOD_mouse_t* **mouse*)

Check for a specific type of event.

Parameters

- **eventMask** – A bit-mask of TCOD_event_t flags.
- **key** – Optional pointer to a *TCOD_key_t* struct.
- **mouse** – Optional pointer to a *TCOD_mouse_t* struct.
- **flush** – This should always be false.

Returns A TCOD_event_t flag showing which event was actually processed.

TCOD_mouse_t **TCOD_mouse_get_status**(void)

Return a copy of the current mouse state.

2.4.3 Keyboard event structure

enum TCOD_key_status_t

Values:

enumerator TCOD_KEY_PRESSED

enumerator TCOD_KEY_RELEASED

struct TCOD_key_t

2.4.4 Key codes

enum TCOD_keycode_t

Values:

enumerator TCODK_NONE

enumerator TCODK_ESCAPE

enumerator TCODK_BACKSPACE

enumerator TCODK_TAB

enumerator TCODK_ENTER

enumerator TCODK_SHIFT

enumerator TCODK_CONTROL

enumerator TCODK_ALT

enumerator TCODK_PAUSE

enumerator TCODK_CAPSLOCK

enumerator TCODK_PAGEUP

enumerator TCODK_PAGEDOWN

enumerator TCODK_END

enumerator TCODK_HOME

enumerator TCODK_UP

enumerator TCODK_LEFT

enumerator TCODK_RIGHT

enumerator TCODK_DOWN

enumerator TCODK_PRINTSCREEN

enumerator TCODK_INSERT

enumerator TCODK_DELETE

enumerator TCODK_LWIN

enumerator TCODK_RWIN

enumerator TCODK_APPS

enumerator TCODK_0

enumerator TCODK_1

```
enumerator TCODK_2
enumerator TCODK_3
enumerator TCODK_4
enumerator TCODK_5
enumerator TCODK_6
enumerator TCODK_7
enumerator TCODK_8
enumerator TCODK_9
enumerator TCODK_KP0
enumerator TCODK_KP1
enumerator TCODK_KP2
enumerator TCODK_KP3
enumerator TCODK_KP4
enumerator TCODK_KP5
enumerator TCODK_KP6
enumerator TCODK_KP7
enumerator TCODK_KP8
enumerator TCODK_KP9
enumerator TCODK_KPADD
enumerator TCODK_KPSUB
enumerator TCODK_KPDIV
enumerator TCODK_KPMUL
enumerator TCODK_KPDEC
enumerator TCODK_KPENTER
enumerator TCODK_F1
enumerator TCODK_F2
enumerator TCODK_F3
enumerator TCODK_F4
enumerator TCODK_F5
enumerator TCODK_F6
enumerator TCODK_F7
enumerator TCODK_F8
enumerator TCODK_F9
enumerator TCODK_F10
enumerator TCODK_F11
enumerator TCODK_F12
```

enumerator TCODK_NUMLOCK

enumerator TCODK_SCROLLLOCK

enumerator TCODK_SPACE

enumerator TCODK_CHAR

enumerator TCODK_TEXT

2.4.5 Mouse event structure

struct TCOD_mouse_t

2.4.6 Events from SDL2

TCOD_event_t tcod::sdl2::**process_event**(**const union** SDL_Event &*in*, *TCOD_key_t* &*out*) **noexcept**

Parse an SDL_Event into a key event and return the relevant TCOD_event_t.

Returns TCOD_EVENT_NONE if the event wasn't keyboard related.

New in version 1.11.

TCOD_event_t **TCOD_sys_process_key_event**(**const union** SDL_Event **in*, *TCOD_key_t* **out*)

Parse an SDL_Event into a key event and return the relevant TCOD_event_t.

Returns TCOD_EVENT_NONE if the event wasn't keyboard related.

New in version 1.11.

TCOD_event_t tcod::sdl2::**process_event**(**const union** SDL_Event &*in*, *TCOD_mouse_t* &*out*) **noexcept**

Parse an SDL_Event into a mouse event and return the relevant TCOD_event_t.

Returns TCOD_EVENT_NONE if the event wasn't mouse related.

New in version 1.11.

TCOD_event_t **TCOD_sys_process_mouse_event**(**const union** SDL_Event **in*, *TCOD_mouse_t* **out*)

Parse an SDL_Event into a mouse event and return the relevant TCOD_event_t.

Returns TCOD_EVENT_NONE if the event wasn't mouse related.

New in version 1.11.

2.5 Using off-screen consoles

2.5.1 Creating and deleting off-screen consoles

TCOD_Console ****TCOD_console_new**(int *w*, int *h*)

Return a new console with a specific number of columns and rows.

Parameters

- **w** – Number of columns.
- **h** – Number of columns.

Returns A pointer to the new console, or NULL on error.

void **TCOD_console_delete**(TCOD_Console *_console_)
Delete a console.

If the console being deleted is the root console, then the display will be uninitialized.

Parameters **con** – A console pointer.

2.5.2 Creating an off-screen console from any .asc/.apf/.xp file

TCOD_console_t **TCOD_console_from_file**(**const** char *_filename_)

2.5.3 Loading an offscreen console from a .asc file

bool **TCOD_console_load_asc**(TCOD_console_t _con_, **const** char *_filename_)

2.5.4 Loading an offscreen console from a .apf file

bool **TCOD_console_load_apf**(TCOD_console_t _con_, **const** char *_filename_)

2.5.5 Saving a console to a .asc file

bool **TCOD_console_save_asc**(TCOD_console_t _con_, **const** char *_filename_)

2.5.6 Saving a console to a .apf file

bool **TCOD_console_save_apf**(TCOD_console_t _con_, **const** char *_filename_)

2.5.7 Working with REXPaint `.xp` files

REXPaint gives special treatment to tiles with a magic pink `{255, 0, 255}` background color. You can processes this effect manually or by setting *`TCOD_console_set_key_color()`* to *TCOD_fuchsia*.

`libtcodpy`.**console_from_xp**(*filename*)

TCOD_console_t **TCOD_console_from_xp**(**const** char *_filename_)
Return a new console loaded from a REXPaint `.xp` file.

Parameters **[in] filename** – A path to the REXPaint file.

Returns A new TCOD_console_t object. New consoles will need to be deleted with a call to :any:`TCOD_console_delete`. Returns NULL on an error.

`libtcodpy`.**console_load_xp**(*con*, *filename*)

bool `TCODConsole`::**loadXp**(**const** char *_filename_)

bool **TCOD_console_load_xp**(TCOD_Console *_con_, **const** char *_filename_)

`libtcodpy`.**console_save_xp**(*con*, *filename*, *compress_level=- 1*)

bool `TCODConsole`::**saveXp**(**const** char *_filename_, int _compress_level_)

bool **TCOD_console_save_xp**(**const** TCOD_Console *con, **const** char *filename, int compress_level)
Save a console as a REXPaint `.xp` file.

The REXPaint format can support a 1:1 copy of a libtcod console.

Parameters

- **[in] con** – The console instance to save.
- **[in] filename** – The filepath to save to.
- **[in] compress_level** – A zlib compression level, from 0 to 9. 1=fast, 6=balanced, 9=slowest, 0=uncompressed.

Returns `true` when the file is saved successfully, or `false` when an issue is detected.

libtcodpy.**console_list_from_xp**(*filename*)

TCOD_list_t **TCOD_console_list_from_xp**(**const** char *filename)
Return a list of consoles from a REXPaint file.

This function can load a REXPaint file with variable layer shapes, which would cause issues for a function like TCOD_console_list_from_xp.

Parameters **[in] filename** – A path to the REXPaint file.

Returns Returns a TCOD_list_t of TCOD_console_t objects. Or NULL on an error. You will need to delete this list and each console individually.

libtcodpy.**console_list_save_xp**(*console_list*, *filename*, *compress_level*)

bool **TCOD_console_list_save_xp**(TCOD_list_t *console_list*, **const** char *filename, int compress_level)
Save a list of consoles to a REXPaint file.

This function can save any number of layers with multiple different sizes.

The REXPaint tool only supports files with up to 9 layers where all layers are the same size.

Parameters

- **[in] console_list** – A TCOD_list_t of TCOD_console_t objects.
- **[in] filename** – Path to save to.
- **[in] compress_level** – zlib compression level.

Returns true on success, false on a failure such as not being able to write to the path provided.

2.5.8 Blitting a console on another one

void **TCOD_console_blit**(**const** TCOD_Console *src, int xSrc, int ySrc, int wSrc, int hSrc, TCOD_Console *dst, int xDst, int yDst, float foreground_alpha, float background_alpha)
Blit from one console to another.

If the source console has a key color, this function will use it.

Changed in version 1.16: Blits can now handle per-cell alpha transparency.

Parameters

- **srcCon** – Pointer to the source console.
- **xSrc** – The left region of the source console to blit from.
- **ySrc** – The top region of the source console to blit from.
- **wSrc** – The width of the region to blit from. If 0 then it will fill to the maximum width.
- **hSrc** – The height of the region to blit from. If 0 then it will fill to the maximum height.
- **dstCon** – Pointer to the destination console.
- **xDst** – The left corner to blit onto the destination console.
- **yDst** – The top corner to blit onto the destination console.
- **foreground_alpha** – Foreground blending alpha.
- **background_alpha** – Background blending alpha.

2.5.9 Define a blit-transparent color

void **TCOD_console_set_key_color**(TCOD_Console *con, TCOD_color_t col)

CHAPTER

THREE

SYSTEM LAYER

Note: These docs are incomplete. Several functions and features are better documented by the older 1.6.4 docs which you can find here.

3.1 Time functions

uint32_t **TCOD_sys_elapsed_milli** (void)

float **TCOD_sys_elapsed_seconds** (void)

void **TCOD_sys_sleep_milli** (uint32_t *val*)

void **TCOD_sys_set_fps** (int *val*)

int **TCOD_sys_get_fps** (void)

float **TCOD_sys_get_last_frame_length** (void)

3.2 Easy screenshots

void **TCOD_sys_save_screenshot** (**const** char **filename*)

3.3 Miscellaneous utilities

struct SDL_Window ***TCOD_sys_get_sdl_window** (void)
Return an SDL_Window pointer if one is in use, returns NULL otherwise.

New in version 1.11.

struct SDL_Renderer ***TCOD_sys_get_sdl_renderer** (void)
Return an SDL_Renderer pointer if one is in use, returns NULL otherwise.

New in version 1.11.

CHAPTER

FOUR

LINE DRAWING TOOLKIT

4.1 Iterator-based line drawing

```
#include <libtcod.h>

void main() {
  TCOD_bresenham_data_t bresenham_data;
  int x=5, y=8;
  TCOD_line_init_mt(x, y, 13, 14, &bresenham_data);
  do {
    printf("%d %d\n", x, y);
  } while (!TCOD_line_step_mt(&x, &y, &bresenham_data));
}
```

struct TCOD_bresenham_data_t
A struct used for computing a bresenham line.

void **TCOD_line_init_mt**(int *xFrom*, int *yFrom*, int *xTo*, int *yTo*, *TCOD_bresenham_data_t* **data*)
Initialize a *TCOD_bresenham_data_t* struct.

After calling this function you use TCOD_line_step_mt to iterate over the individual points on the line.

Parameters

- **xFrom** – The starting x position.
- **yFrom** – The starting y position.
- **xTo** – The ending x position.
- **yTo** – The ending y position.
- **data** – Pointer to a *TCOD_bresenham_data_t* struct.

bool **TCOD_line_step_mt**(int **xCur*, int **yCur*, *TCOD_bresenham_data_t* **data*)
Get the next point on a line, returns true once the line has ended.

The starting point is excluded by this function. After the ending point is reached, the next call will return true.

Parameters

- **xCur** – An int pointer to fill with the next x position.
- **yCur** – An int pointer to fill with the next y position.
- **data** – Pointer to a initialized *TCOD_bresenham_data_t* struct.

Returns true after the ending point has been reached.

4.2 Callback-based line drawing

```
#include <libtcod.h>

void main() {
  bool my_listener(int x,int y) {
    printf("%d %d\n", x, y);
    return true;
  }
  TCOD_line(5, 8, 13, 4, my_listener);
}
```

typedef bool (***TCOD_line_listener_t**) (int x, int y)
A callback to be passed to TCOD_line.

The points given to the callback include both the starting and ending positions.

Parameters

- **x** –
- **y** –

Returns As long as this callback returns true it will be called with the next x,y point on the line.

bool **TCOD_line** (int *xFrom*, int *yFrom*, int *xTo*, int *yTo*, *TCOD_line_listener_t* *listener*)
Iterate over a line using a callback.

Changed in version 1.6.6: This function is now reentrant.

Parameters

- **xo** – The origin x position.
- **yo** – The origin y position.
- **xd** – The destination x position.
- **yd** – The destination y position.
- **listener** – A TCOD_line_listener_t callback.

Returns true if the line was completely exhausted by the callback.

class tcod::**BresenhamLine**
Encapsulates a Bresenham line drawing algorithm.

This class is provisional.

Public Functions

inline BresenhamLine (int *xFrom*, int *yFrom*, int *xTo*, int *yTo*)
Initializes the object to draw a line from `xFrom`, `yFrom` to `xTo`, `yTo`.

inline bool **step** (int &*x*, int &*y*)
Steps through each cell from the start to the end coordinates.

The input variables `x`, `y` are passed in by reference and updated at each step of the line.

inline bool `tcod::bresenham_line` (int *xFrom*, int *yFrom*, int *xTo*, int *yTo*, **const** std::function<bool) int, int > &*callback*Draw a Bresenham line, passing the indexes of the line to `callback`.

This function is provisional.

4.3 Deprecated functions

bool **TCOD_line_mt** (int *xFrom*, int *yFrom*, int *xTo*, int *yTo*, *TCOD_line_listener_t* *listener*, *TCOD_bresenham_data_t* **data*)
Iterate over a line using a callback.

Deprecated since version 1.6.6: The *data* parameter for this call is redundant, you should call *TCOD_line()* instead.

Parameters

- **xo** – The origin x position.
- **yo** – The origin y position.
- **xd** – The destination x position.
- **yd** – The destination y position.
- **listener** – A TCOD_line_listener_t callback.
- **data** – Pointer to a *TCOD_bresenham_data_t* struct.

Returns true if the line was completely exhausted by the callback.

void **TCOD_line_init** (int *xFrom*, int *yFrom*, int *xTo*, int *yTo*)
Initialize a line using a global state.

Deprecated since version 1.6.6: This function is not reentrant and will fail if a new line is started before the last is finished processing.

Use *TCOD_line_init_mt()* instead.

Parameters

- **xFrom** – The starting x position.
- **yFrom** – The starting y position.

- **xTo** – The ending x position.
- **yTo** – The ending y position.

bool **TCOD_line_step** (int *xCur, int *yCur)

advance one step.

returns true if we reach destination

advance one step.

The starting point is excluded by this function. After the ending point is reached, the next call will return true.

Deprecated since version 1.6.6: This function is not reentrant and will fail if a new line is started before the last is finished processing.

Use *TCOD_line_step_mt()* instead.

Parameters

- **xCur** – An int pointer to fill with the next x position.
- **yCur** – An int pointer to fill with the next y position.

Returns true once the ending point has been reached.

CHAPTER

FIVE

UPGRADING

5.1 1.5.x -> 1.6.x

The largest and most influential change to libtcod, between versions 1.5.2 and 1.6.0, was the move to replace SDL with SDL2. SDL2 made many extensive changes to concepts used in SDL. Only one of these changes, the separation of text and key events, required a change in the libtcod API requiring users to update their code in the process of updating the version of libtcod they use.

When a user presses a key, they may be pressing `SHIFT` and =. On some keyboards, depending on the user's language and location, this may show + on the screen. On other user's keyboards, who knows what it may show on screen. SDL2 changes the way "the text which is displayed on the user's screen" is sent in key events. This means that the key event for `SHIFT` and = will be what happens for presses of both + and = (for user's with applicable keyboards), and there will be a new text event that happens with the displayed +.

5.1.1 In libtcod 1.5.x

SDL would when sending key events, provide the unicode character for the key event, ready for use. This meant that if the user happened to be using a British keyboard (or any that are similarly laid out), and pressed `SHIFT` and =, the event would be for the character +.

Listing 1: C / C++

```
if (key->c == '+') {
    /* Handle any key that displays a plus. */
}
```

Listing 2: Python

```
if key.c == "+":
    pass # Handle any key that displays a plus.
```

5.1.2 In libtcod 1.6.x

With SDL2, the raw key-presses still occur, but they are fundamentally linked to the keyboard of the user. Now there will still be an event where it says `SHIFT` and = are pressed, but the event will always be for the unmodified character =. The unicode text arrives in a new kind of event, and getting it requires explicitly checking that the event is the new text event, and then looking for the value in the relevant `text` field for the language being used.

Listing 3: C / C++

```
if (key->vk == TCODK_TEXT)
    if (key.text[0] == '+') {
        ; /* Handle any key that displays a plus. */
    }
```

Listing 4: Python

```
if key.vk == libtcod.KEY_TEXT:
    if key.text == "+":
        pass # Handle any key that displays a plus.
```

5.1.3 Still confused?

Run your code from a terminal or DOS window and print out the event attributes/fields and look at what is going on. Have your code print out the modifiers, the keycode, the character, the text, and then run it and try pressing some keys. It will be much faster than posting "I don't understand" or "Can someone explain" somewhere and waiting for a response.

CHAPTER

SIX

GETTING STARTED

Here is a simple example of setting up a libtcod context in C++ without using deprecated functions.

```
#include <libtcod.h>
#include <SDL2.h>

int main(int argc, char* argv[]) {
  tcod::ConsolePtr console = tcod::new_console(80, 25);  // Main console.

  // Configure the context.
  TCOD_ContextParams params = {};
  params.tcod_version = TCOD_COMPILEDVERSION;  // This is required.
  params.columns = console->w;  // Derive the window size from the console size.
  params.rows = console->h;
  params.sdl_window_flags = SDL_WINDOW_RESIZABLE;
  params.vsync = true;
  params.argc = argc;  // This allows some user-control of the context.
  params.argv = argv;
  tcod::ContextPtr context = tcod::new_context(params);

  while (1) {  // Game loop.
    TCOD_console_clear(console.get());
    tcod::print(*console, 0, 0, "Hello World", nullptr, nullptr, TCOD_BKGND_NONE,␣
↪TCOD_LEFT);
    context->present(*console);  // Updates the visible display.

    SDL_Event event;
    SDL_WaitEvent(&event);
    switch (event.type) {
      case SDL_QUIT:
        return 0;  // Exit.
    }
  }
}
```

INDEX

B

built-in function
    libtcodpy.console_from_xp(), 28
    libtcodpy.console_list_from_xp(), 29
    libtcodpy.console_list_save_xp(), 29
    libtcodpy.console_load_xp(), 28
    libtcodpy.console_save_xp(), 28

L

libtcodpy.console_from_xp()
    built-in function, 28
libtcodpy.console_list_from_xp()
    built-in function, 29
libtcodpy.console_list_save_xp()
    built-in function, 29
libtcodpy.console_load_xp()
    built-in function, 28
libtcodpy.console_save_xp()
    built-in function, 28

T

tcod::bresenham_line (*C++ function*), 35
tcod::BresenhamLine (*C++ class*), 34
tcod::BresenhamLine::BresenhamLine (*C++ function*), 35
tcod::BresenhamLine::step (*C++ function*), 35
tcod::sdl2::process_event (*C++ function*), 27
TCOD_alignment_t (*C++ enum*), 16
TCOD_alignment_t::TCOD_CENTER (*C++ enumerator*), 16
TCOD_alignment_t::TCOD_LEFT (*C++ enumerator*), 16
TCOD_alignment_t::TCOD_RIGHT (*C++ enumerator*), 16
TCOD_bkgnd_flag_t (*C++ enum*), 15
TCOD_bkgnd_flag_t::TCOD_BKGND_ADD (*C++ enumerator*), 16
TCOD_bkgnd_flag_t::TCOD_BKGND_ADDA (*C++ enumerator*), 16
TCOD_bkgnd_flag_t::TCOD_BKGND_ALPH (*C++ enumerator*), 16
TCOD_bkgnd_flag_t::TCOD_BKGND_BURN (*C++ enumerator*), 16
TCOD_bkgnd_flag_t::TCOD_BKGND_COLOR_BURN (*C++ enumerator*), 16
TCOD_bkgnd_flag_t::TCOD_BKGND_COLOR_DODGE (*C++ enumerator*), 16
TCOD_bkgnd_flag_t::TCOD_BKGND_DARKEN (*C++ enumerator*), 15
TCOD_bkgnd_flag_t::TCOD_BKGND_DEFAULT (*C++ enumerator*), 16
TCOD_bkgnd_flag_t::TCOD_BKGND_LIGHTEN (*C++ enumerator*), 15
TCOD_bkgnd_flag_t::TCOD_BKGND_MULTIPLY (*C++ enumerator*), 15
TCOD_bkgnd_flag_t::TCOD_BKGND_NONE (*C++ enumerator*), 15
TCOD_bkgnd_flag_t::TCOD_BKGND_OVERLAY (*C++ enumerator*), 16
TCOD_bkgnd_flag_t::TCOD_BKGND_SCREEN (*C++ enumerator*), 16
TCOD_bkgnd_flag_t::TCOD_BKGND_SET (*C++ enumerator*), 15
TCOD_bresenham_data_t (*C++ struct*), 33
TCOD_chars_t (*C++ enum*), 21
TCOD_chars_t::TCOD_CHAR_ARROW2_E (*C++ enumerator*), 21
TCOD_chars_t::TCOD_CHAR_ARROW2_N (*C++ enumerator*), 21
TCOD_chars_t::TCOD_CHAR_ARROW2_S (*C++ enumerator*), 21
TCOD_chars_t::TCOD_CHAR_ARROW2_W (*C++ enumerator*), 21
TCOD_chars_t::TCOD_CHAR_ARROW_E (*C++ enumerator*), 21
TCOD_chars_t::TCOD_CHAR_ARROW_N (*C++ enumerator*), 21
TCOD_chars_t::TCOD_CHAR_ARROW_S (*C++ enumerator*), 21
TCOD_chars_t::TCOD_CHAR_ARROW_W (*C++ enumerator*), 21
TCOD_chars_t::TCOD_CHAR_BLOCK1 (*C++ enumerator*), 21

TCOD_chars_t::TCOD_CHAR_BLOCK2 (C++ enumerator), 21
TCOD_chars_t::TCOD_CHAR_BLOCK3 (C++ enumerator), 21
TCOD_chars_t::TCOD_CHAR_BULLET (C++ enumerator), 22
TCOD_chars_t::TCOD_CHAR_BULLET_INV (C++ enumerator), 22
TCOD_chars_t::TCOD_CHAR_BULLET_SQUARE (C++ enumerator), 23
TCOD_chars_t::TCOD_CHAR_CENT (C++ enumerator), 22
TCOD_chars_t::TCOD_CHAR_CHECKBOX_SET (C++ enumerator), 22
TCOD_chars_t::TCOD_CHAR_CHECKBOX_UNSET (C++ enumerator), 22
TCOD_chars_t::TCOD_CHAR_CLUB (C++ enumerator), 22
TCOD_chars_t::TCOD_CHAR_COPYRIGHT (C++ enumerator), 22
TCOD_chars_t::TCOD_CHAR_CROSS (C++ enumerator), 21
TCOD_chars_t::TCOD_CHAR_CURRENCY (C++ enumerator), 23
TCOD_chars_t::TCOD_CHAR_DARROW_H (C++ enumerator), 21
TCOD_chars_t::TCOD_CHAR_DARROW_V (C++ enumerator), 22
TCOD_chars_t::TCOD_CHAR_DCROSS (C++ enumerator), 21
TCOD_chars_t::TCOD_CHAR_DHLINE (C++ enumerator), 21
TCOD_chars_t::TCOD_CHAR_DIAMOND (C++ enumerator), 22
TCOD_chars_t::TCOD_CHAR_DIVISION (C++ enumerator), 23
TCOD_chars_t::TCOD_CHAR_DNE (C++ enumerator), 21
TCOD_chars_t::TCOD_CHAR_DNW (C++ enumerator), 21
TCOD_chars_t::TCOD_CHAR_DSE (C++ enumerator), 21
TCOD_chars_t::TCOD_CHAR_DSW (C++ enumerator), 21
TCOD_chars_t::TCOD_CHAR_DTEEE (C++ enumerator), 21
TCOD_chars_t::TCOD_CHAR_DTEEN (C++ enumerator), 21
TCOD_chars_t::TCOD_CHAR_DTEES (C++ enumerator), 21
TCOD_chars_t::TCOD_CHAR_DTEEW (C++ enumerator), 21
TCOD_chars_t::TCOD_CHAR_DVLINE (C++ enumerator), 21
TCOD_chars_t::TCOD_CHAR_EXCLAM_DOUBLE (C++ enumerator), 22
TCOD_chars_t::TCOD_CHAR_FEMALE (C++ enumerator), 22
TCOD_chars_t::TCOD_CHAR_FUNCTION (C++ enumerator), 22
TCOD_chars_t::TCOD_CHAR_GRADE (C++ enumerator), 23
TCOD_chars_t::TCOD_CHAR_HALF (C++ enumerator), 22
TCOD_chars_t::TCOD_CHAR_HEART (C++ enumerator), 22
TCOD_chars_t::TCOD_CHAR_HLINE (C++ enumerator), 21
TCOD_chars_t::TCOD_CHAR_LIGHT (C++ enumerator), 22
TCOD_chars_t::TCOD_CHAR_MALE (C++ enumerator), 22
TCOD_chars_t::TCOD_CHAR_MULTIPLICATION (C++ enumerator), 22
TCOD_chars_t::TCOD_CHAR_NE (C++ enumerator), 21
TCOD_chars_t::TCOD_CHAR_NOTE (C++ enumerator), 22
TCOD_chars_t::TCOD_CHAR_NOTE_DOUBLE (C++ enumerator), 22
TCOD_chars_t::TCOD_CHAR_NW (C++ enumerator), 21
TCOD_chars_t::TCOD_CHAR_ONE_QUARTER (C++ enumerator), 22
TCOD_chars_t::TCOD_CHAR_PILCROW (C++ enumerator), 22
TCOD_chars_t::TCOD_CHAR_POUND (C++ enumerator), 22
TCOD_chars_t::TCOD_CHAR_POW1 (C++ enumerator), 23
TCOD_chars_t::TCOD_CHAR_POW2 (C++ enumerator), 23
TCOD_chars_t::TCOD_CHAR_POW3 (C++ enumerator), 23
TCOD_chars_t::TCOD_CHAR_RADIO_SET (C++ enumerator), 22
TCOD_chars_t::TCOD_CHAR_RADIO_UNSET (C++ enumerator), 22
TCOD_chars_t::TCOD_CHAR_RESERVED (C++ enumerator), 22
TCOD_chars_t::TCOD_CHAR_SE (C++ enumerator), 21
TCOD_chars_t::TCOD_CHAR_SECTION (C++ enumerator), 22
TCOD_chars_t::TCOD_CHAR_SMILIE (C++ enumerator), 22
TCOD_chars_t::TCOD_CHAR_SMILIE_INV (C++ enumerator), 22

TCOD_chars_t::TCOD_CHAR_SPADE (C++ enumerator), 22
TCOD_chars_t::TCOD_CHAR_SUBP_DIAG (C++ enumerator), 22
TCOD_chars_t::TCOD_CHAR_SUBP_E (C++ enumerator), 22
TCOD_chars_t::TCOD_CHAR_SUBP_N (C++ enumerator), 22
TCOD_chars_t::TCOD_CHAR_SUBP_NE (C++ enumerator), 22
TCOD_chars_t::TCOD_CHAR_SUBP_NW (C++ enumerator), 22
TCOD_chars_t::TCOD_CHAR_SUBP_SE (C++ enumerator), 22
TCOD_chars_t::TCOD_CHAR_SUBP_SW (C++ enumerator), 22
TCOD_chars_t::TCOD_CHAR_SW (C++ enumerator), 21
TCOD_chars_t::TCOD_CHAR_TEEE (C++ enumerator), 21
TCOD_chars_t::TCOD_CHAR_TEEN (C++ enumerator), 21
TCOD_chars_t::TCOD_CHAR_TEES (C++ enumerator), 21
TCOD_chars_t::TCOD_CHAR_TEEW (C++ enumerator), 21
TCOD_chars_t::TCOD_CHAR_THREE_QUARTERS (C++ enumerator), 23
TCOD_chars_t::TCOD_CHAR_UMLAUT (C++ enumerator), 23
TCOD_chars_t::TCOD_CHAR_VLINE (C++ enumerator), 21
TCOD_chars_t::TCOD_CHAR_YEN (C++ enumerator), 23
TCOD_color_add (C++ function), 4
TCOD_color_equals (C++ function), 3
TCOD_color_gen_map (C++ function), 7
TCOD_color_get_HSV (C++ function), 5
TCOD_color_get_hue (C++ function), 6
TCOD_color_get_saturation (C++ function), 6
TCOD_color_get_value (C++ function), 6
TCOD_color_lerp (C++ function), 4
TCOD_color_multiply (C++ function), 4
TCOD_color_multiply_scalar (C++ function), 4
TCOD_color_scale_HSV (C++ function), 6
TCOD_color_set_HSV (C++ function), 5
TCOD_color_set_hue (C++ function), 5
TCOD_color_set_saturation (C++ function), 5
TCOD_color_set_value (C++ function), 5
TCOD_color_shift_hue (C++ function), 6
TCOD_color_subtract (C++ function), 4
TCOD_console_blit (C++ function), 29
TCOD_console_check_for_keypress (C++ function), 24
TCOD_console_clear (C++ function), 13
TCOD_console_credits (C++ function), 13
TCOD_console_credits_render (C++ function), 13
TCOD_console_credits_reset (C++ function), 13
TCOD_console_delete (C++ function), 28
TCOD_console_flush (C++ function), 23
TCOD_console_from_file (C++ function), 28
TCOD_console_from_xp (C++ function), 28
TCOD_console_get_alignment (C++ function), 16
TCOD_console_get_background_flag (C++ function), 20
TCOD_console_get_char (C++ function), 19
TCOD_console_get_char_background (C++ function), 20
TCOD_console_get_char_foreground (C++ function), 20
TCOD_console_get_default_background (C++ function), 20
TCOD_console_get_default_foreground (C++ function), 20
TCOD_console_get_fade (C++ function), 20
TCOD_console_get_fading_color (C++ function), 20
TCOD_console_get_height (C++ function), 19
TCOD_console_get_height_rect (C++ function), 17
TCOD_console_get_height_rect_fmt (C++ function), 18
TCOD_console_get_height_rect_utf (C++ function), 19
TCOD_console_get_width (C++ function), 19
TCOD_console_has_mouse_focus (C++ function), 13
TCOD_console_hline (C++ function), 15
TCOD_console_init_root (C++ function), 10
TCOD_console_is_active (C++ function), 13
TCOD_console_is_fullscreen (C++ function), 12
TCOD_console_is_key_pressed (C++ function), 24
TCOD_console_is_window_closed (C++ function), 13
TCOD_console_list_from_xp (C++ function), 29
TCOD_console_list_save_xp (C++ function), 29
TCOD_console_load_apf (C++ function), 28
TCOD_console_load_asc (C++ function), 28
TCOD_console_load_xp (C++ function), 28
TCOD_console_map_ascii_code_to_font (C++ function), 12
TCOD_console_map_ascii_codes_to_font (C++ function), 12

TCOD_console_map_string_to_font (C++ function), 12
TCOD_console_new (C++ function), 27
TCOD_console_print (C++ function), 16
TCOD_console_print_ex (C++ function), 16
TCOD_console_print_ex_utf (C++ function), 19
TCOD_console_print_frame (C++ function), 15
TCOD_console_print_rect (C++ function), 17
TCOD_console_print_rect_ex (C++ function), 17
TCOD_console_print_rect_ex_utf (C++ function), 19
TCOD_console_print_rect_utf (C++ function), 19
TCOD_console_print_utf (C++ function), 19
TCOD_console_printf (C++ function), 18
TCOD_console_printf_ex (C++ function), 18
TCOD_console_printf_frame (C++ function), 19
TCOD_console_printf_rect (C++ function), 18
TCOD_console_printf_rect_ex (C++ function), 18
TCOD_console_put_char (C++ function), 13
TCOD_console_put_char_ex (C++ function), 14
TCOD_console_rect (C++ function), 14
TCOD_console_save_apf (C++ function), 28
TCOD_console_save_asc (C++ function), 28
TCOD_console_save_xp (C++ function), 28
TCOD_console_set_alignment (C++ function), 16
TCOD_console_set_background_flag (C++ function), 13
TCOD_console_set_char (C++ function), 14
TCOD_console_set_char_background (C++ function), 14
TCOD_console_set_char_foreground (C++ function), 14
TCOD_console_set_custom_font (C++ function), 11
TCOD_console_set_default_background (C++ function), 13
TCOD_console_set_default_foreground (C++ function), 13
TCOD_console_set_fade (C++ function), 20
TCOD_console_set_fullscreen (C++ function), 12
TCOD_console_set_key_color (C++ function), 30
TCOD_console_set_window_title (C++ function), 13
TCOD_console_vline (C++ function), 15
TCOD_console_wait_for_keypress (C++ function), 23
TCOD_font_flags_t (C++ enum), 11
TCOD_font_flags_t::TCOD_FONT_LAYOUT_ASCII_INCOL (C++ enumerator), 11
TCOD_font_flags_t::TCOD_FONT_LAYOUT_ASCII_INROW (C++ enumerator), 11
TCOD_font_flags_t::TCOD_FONT_LAYOUT_CP437 (C++ enumerator), 11
TCOD_font_flags_t::TCOD_FONT_LAYOUT_TCOD (C++ enumerator), 11
TCOD_font_flags_t::TCOD_FONT_TYPE_GRAYSCALE (C++ enumerator), 11
TCOD_font_flags_t::TCOD_FONT_TYPE_GREYSCALE (C++ enumerator), 11
TCOD_key_status_t (C++ enum), 25
TCOD_key_status_t::TCOD_KEY_PRESSED (C++ enumerator), 25
TCOD_key_status_t::TCOD_KEY_RELEASED (C++ enumerator), 25
TCOD_key_t (C++ struct), 25
TCOD_keycode_t (C++ enum), 25
TCOD_keycode_t::TCODK_0 (C++ enumerator), 25
TCOD_keycode_t::TCODK_1 (C++ enumerator), 25
TCOD_keycode_t::TCODK_2 (C++ enumerator), 25
TCOD_keycode_t::TCODK_3 (C++ enumerator), 26
TCOD_keycode_t::TCODK_4 (C++ enumerator), 26
TCOD_keycode_t::TCODK_5 (C++ enumerator), 26
TCOD_keycode_t::TCODK_6 (C++ enumerator), 26
TCOD_keycode_t::TCODK_7 (C++ enumerator), 26
TCOD_keycode_t::TCODK_8 (C++ enumerator), 26
TCOD_keycode_t::TCODK_9 (C++ enumerator), 26
TCOD_keycode_t::TCODK_ALT (C++ enumerator), 25
TCOD_keycode_t::TCODK_APPS (C++ enumerator), 25
TCOD_keycode_t::TCODK_BACKSPACE (C++ enumerator), 25
TCOD_keycode_t::TCODK_CAPSLOCK (C++ enumerator), 25
TCOD_keycode_t::TCODK_CHAR (C++ enumerator), 27
TCOD_keycode_t::TCODK_CONTROL (C++ enumerator), 25
TCOD_keycode_t::TCODK_DELETE (C++ enumerator), 25
TCOD_keycode_t::TCODK_DOWN (C++ enumerator), 25

TCOD_keycode_t::TCODK_END (C++ enumerator), 25
TCOD_keycode_t::TCODK_ENTER (C++ enumerator), 25
TCOD_keycode_t::TCODK_ESCAPE (C++ enumerator), 25
TCOD_keycode_t::TCODK_F1 (C++ enumerator), 26
TCOD_keycode_t::TCODK_F10 (C++ enumerator), 26
TCOD_keycode_t::TCODK_F11 (C++ enumerator), 26
TCOD_keycode_t::TCODK_F12 (C++ enumerator), 26
TCOD_keycode_t::TCODK_F2 (C++ enumerator), 26
TCOD_keycode_t::TCODK_F3 (C++ enumerator), 26
TCOD_keycode_t::TCODK_F4 (C++ enumerator), 26
TCOD_keycode_t::TCODK_F5 (C++ enumerator), 26
TCOD_keycode_t::TCODK_F6 (C++ enumerator), 26
TCOD_keycode_t::TCODK_F7 (C++ enumerator), 26
TCOD_keycode_t::TCODK_F8 (C++ enumerator), 26
TCOD_keycode_t::TCODK_F9 (C++ enumerator), 26
TCOD_keycode_t::TCODK_HOME (C++ enumerator), 25
TCOD_keycode_t::TCODK_INSERT (C++ enumerator), 25
TCOD_keycode_t::TCODK_KP0 (C++ enumerator), 26
TCOD_keycode_t::TCODK_KP1 (C++ enumerator), 26
TCOD_keycode_t::TCODK_KP2 (C++ enumerator), 26
TCOD_keycode_t::TCODK_KP3 (C++ enumerator), 26
TCOD_keycode_t::TCODK_KP4 (C++ enumerator), 26
TCOD_keycode_t::TCODK_KP5 (C++ enumerator), 26
TCOD_keycode_t::TCODK_KP6 (C++ enumerator), 26
TCOD_keycode_t::TCODK_KP7 (C++ enumerator), 26
TCOD_keycode_t::TCODK_KP8 (C++ enumerator), 26
TCOD_keycode_t::TCODK_KP9 (C++ enumerator), 26
TCOD_keycode_t::TCODK_KPADD (C++ enumerator), 26
TCOD_keycode_t::TCODK_KPDEC (C++ enumerator), 26
TCOD_keycode_t::TCODK_KPDIV (C++ enumerator), 26
TCOD_keycode_t::TCODK_KPENTER (C++ enumerator), 26
TCOD_keycode_t::TCODK_KPMUL (C++ enumerator), 26
TCOD_keycode_t::TCODK_KPSUB (C++ enumerator), 26
TCOD_keycode_t::TCODK_LEFT (C++ enumerator), 25
TCOD_keycode_t::TCODK_LWIN (C++ enumerator), 25
TCOD_keycode_t::TCODK_NONE (C++ enumerator), 25
TCOD_keycode_t::TCODK_NUMLOCK (C++ enumerator), 26
TCOD_keycode_t::TCODK_PAGEDOWN (C++ enumerator), 25
TCOD_keycode_t::TCODK_PAGEUP (C++ enumerator), 25
TCOD_keycode_t::TCODK_PAUSE (C++ enumerator), 25
TCOD_keycode_t::TCODK_PRINTSCREEN (C++ enumerator), 25
TCOD_keycode_t::TCODK_RIGHT (C++ enumerator), 25
TCOD_keycode_t::TCODK_RWIN (C++ enumerator), 25
TCOD_keycode_t::TCODK_SCROLLLOCK (C++ enumerator), 27
TCOD_keycode_t::TCODK_SHIFT (C++ enumerator), 25
TCOD_keycode_t::TCODK_SPACE (C++ enumerator), 27
TCOD_keycode_t::TCODK_TAB (C++ enumerator), 25
TCOD_keycode_t::TCODK_TEXT (C++ enumerator), 27
TCOD_keycode_t::TCODK_UP (C++ enumerator), 25
TCOD_line (C++ function), 34
TCOD_line_init (C++ function), 35
TCOD_line_init_mt (C++ function), 33
TCOD_line_listener_t (C++ type), 34
TCOD_line_mt (C++ function), 35
TCOD_line_step (C++ function), 36
TCOD_line_step_mt (C++ function), 33
TCOD_mouse_get_status (C++ function), 24
TCOD_mouse_t (C++ struct), 27
TCOD_quit (C++ function), 10

TCOD_renderer_t (*C++ enum*), 9
TCOD_renderer_t::TCOD_NB_RENDERERS (*C++ enumerator*), 10
TCOD_renderer_t::TCOD_RENDERER_GLSL (*C++ enumerator*), 9
TCOD_renderer_t::TCOD_RENDERER_OPENGL (*C++ enumerator*), 9
TCOD_renderer_t::TCOD_RENDERER_OPENGL2 (*C++ enumerator*), 9
TCOD_renderer_t::TCOD_RENDERER_SDL (*C++ enumerator*), 9
TCOD_renderer_t::TCOD_RENDERER_SDL2 (*C++ enumerator*), 9
TCOD_sys_accumulate_console (*C++ function*), 23
TCOD_sys_check_for_event (*C++ function*), 24
TCOD_sys_elapsed_milli (*C++ function*), 31
TCOD_sys_elapsed_seconds (*C++ function*), 31
TCOD_sys_get_fps (*C++ function*), 31
TCOD_sys_get_last_frame_length (*C++ function*), 31
TCOD_sys_get_sdl_renderer (*C++ function*), 31
TCOD_sys_get_sdl_window (*C++ function*), 31
TCOD_sys_process_key_event (*C++ function*), 27
TCOD_sys_process_mouse_event (*C++ function*), 27
TCOD_sys_save_screenshot (*C++ function*), 31
TCOD_sys_set_fps (*C++ function*), 31
TCOD_sys_sleep_milli (*C++ function*), 31
TCOD_sys_wait_for_event (*C++ function*), 23
TCODConsole::loadXp (*C++ function*), 28
TCODConsole::saveXp (*C++ function*), 28