# Table of Contents

1 Introduction .................................................. 1  
   1.1 Quick Start ............................................... 1  
   1.2 Naming Conventions ................................... 2  
      1.2.1 Renaming C Functions ............................... 2  
      1.2.2 Enums and Constants ............................... 2  
      1.2.3 Create and Make .................................... 3  
   1.3 Uniform Vectors ......................................... 3  
   1.4 Limitations ............................................... 4  

2 General SDL ..................................................... 5  

3 Video .......................................................... 6  
   3.1 Rectangles ............................................... 6  
   3.2 Colors .................................................... 7  
   3.3 Windowing System Interaction ........................... 9  
   3.4 Surface .................................................. 10  
   3.5 Misc Surface Operations ................................ 11  

4 Events .......................................................... 13  
   4.1 Activity .................................................. 13  
   4.2 Keys ....................................................... 13  
   4.3 Motions ................................................... 14  
   4.4 Buttons ................................................... 14  
   4.5 Joysticks .................................................. 15  
   4.6 Resizes ................................................... 17  
   4.7 Misc ....................................................... 17  

5 Joystick .......................................................... 19  

6 CDROM .......................................................... 21  

7 OpenGL .......................................................... 23  

8 TrueType ........................................................ 24  

9 Audio ............................................................. 25  

10 SDL gfx by Andreas Schiffler ................................. 29  
   10.1 Graphics Primitives ..................................... 29  
   10.2 Rotation / Zooming ....................................... 30  
   10.3 Managing Frame Rate .................................... 31  
   10.4 RGBA Extras .............................................. 31  
   10.5 Image Filtering .......................................... 32
1 Introduction

The (sdl *) modules are an interface to the SDL (Simple Direct Media Layer) library. The goal is to provide both a clean and direct interface to the lowest level SDL, while extending with higher level concepts where useful, such as default arguments and functional-style application of graphics routines. Several SDL add-on libraries have been wrapped and included with Guile-SDL, including SDL_image (for loading multiple image formats), SDL_ttf (for rendering true type fonts), SDL_mixer (for playing/mixing different audio formats), and SDL_rotozoom (for rotating and scaling images). In addition, some low-level 2D graphics primitives have been provided.

1.1 Quick Start

To whet your appetite, and hopefully get you excited about the ease and flexibility of programming with Guile-SDL, we begin with a simple example. The following program is a simple image browser. You can cycle through images by using space, n or right to go forward, backspace, p or left to go backwards, and escape or q to quit.

```scheme
;; load the SDL module and some useful SRFIs
(use-modules ((sdl sdl) #:prefix SDL:)
  (srfi srfi-1)
  (srfi srfi-2))

;; initialize the video subsystem
(SDL:init 'video)

;; directory to search for images in
(define image-dir "/usr/share/pixmaps/")

;; utility to test if a path is a directory
(define (file? f)
  (let* ((stats (stat f))
         (type (stat:type stats)))
    (eq? type 'regular)))

;; build a ring of image file names
(define image-ring
  (let ((dir (opendir image-dir)))
    (letrec ((D (lambda (ls)
                  (let ((file (readdir dir)))
                    (if (eof-object? file)
                        (begin (closedir dir) ls)
                        (D (cons (string-append image-dir file) ls)))))))
      (apply circular-list (reverse (filter file? (D '()))))))

;; functions to cycle through the ring
(define (next-image)
  (let ((next (car image-ring)))
    (set! image-ring (cdr image-ring)
      next))

(define (prev-image)
Chapter 1: Introduction

1.2 Naming Conventions

The most important thing to learning a wrapped library for a programming language, assuming you know the language and the library, is to know the naming conventions. Then you can begin programming without having to look up the exact function reference (available in the rest of this document).

1.2.1 Renaming C Functions

As with standard guile naming conventions, all names are converted to lower-case, and underscores are replaced with hyphens. Functions that modify one or more arguments have an exclamation point (!) appended, and functions which ask a question and return a boolean value have a question mark (?) appended.

1.2.2 Enums and Constants

SDL enumerated types and constants are passed and returned as symbols, thus enforcing their "constant" nature and for ease of use in case statements. Flags, such as the SDL initialization flags and video surface flags, are treated as lists of symbols, each constant in the flag group that you would or together in C code becoming a symbol in the list.
Some of these symbols retain their exact C names, while others are adapted to better fit Scheme (mostly by removing the ‘SDL_’ prefix, changing underscores to hyphens, downcasing, and inserting a hyphen between “words”).

A particular set of enums is called an enumstash. Likewise flagstash for flags.

You can use kotk to examine the enums and flags encapsulated by these respectively typed objects. You can also use integers where enums/flags are expected, and can convert between the symbol and numeric value with enum->number, number->enum, flags->number and number->flags.

The conversion procs all take stash as the first argument, a symbol that identifies the particular set of enums/flags. For backward compatibility, stash may also be such an object, but this support will be removed after 2013-12-31, when those objects are to be fully internalized.

kotk [name]  [Procedure]
Return the contents of stash name (a symbol), as an alist with symbolic keys, integer values.
If name is omitted, the keys are the names of the all the enum- and flagstashes, and the values have the form:

(N TYPE)

where n is the count of symbols in that stash, and type is a symbol: enums or flags.

enum->number stash symbol  [Procedure]
Return the number in stash associated with symbol.

number->enum stash number  [Procedure]
Return the symbol associated with number, or #f if it does not belong to stash.

flags->number stash flags  [Procedure]
Use stash to convert flags to a number. flags is a list of symbols; or #f, which is taken as the empty list; or #t, which is taken as the list of all possible symbols in stash.

number->flags stash number  [Procedure]
Use stash to convert number to a list of symbols. If the flags in stash are not sufficient to decode number, the first element of the list is the numeric remainder.

Conversion from symbols to numbers (including enum->number and flags->number) throws an error with key non-member-symbol if the specified symbol is not a member of the respective enumstash or flagstash.

1.2.3 Create and Make

The standard SDL prefix for creating a new instance of a type is create. The standard Guile prefix is make. Wherever an SDL function uses the create prefix we will keep it. Object creation functions unique to Guile, such as make-rect, will use make as a prefix. In addition, we will sometimes introduce higher-level creation functions, such as make-surface, which is a wrapper to create-rgb-surface which provides useful default values from the current screen information.

1.3 Uniform Vectors

Some procedures take one or more uniform vector arguments, as specified in SRFI 4 (see Chapter 3 [Video], page 6, see Chapter 10 [SDLgfx], page 29). The specific type of vector is one of u8, u16, s16, where u or s stands for “unsigned” or “signed”, respectively, and the rest the number of bits.
1.4 Limitations

There are some known problems with Guile-SDL modules. This section attempts to make them well-known, if not well-liked...

- API in flux
  Since Guile-SDL is in alpha stage, its interfaces are not stable. Specifically, module names, the contents of modules, procedure names, procedure behavior: all these can change at any time up until the 1.0 release. C’est la vie.

- no logo
  How can any self-respecting package of bindings for libsdl not have a flashy, animated logo? Bonus points for suitable accompanying sound blurb.

- threading picture unclear
  Where do threads fit in if at all? Why doesn’t the Guile-SDL maintainer learn all about threads, fix guile-1.4.x to support that and then arrange for Guile-SDL to DTRT? Questions questions...

- [your gripes here]
2 General SDL

The external representation of a Pixel Format object is:

```
#<SDL-Pixel-Format palette depth ck a chan>
```

where *palette* is the number of colors in the palette, or ‘-1’ if no palette is available; *depth* is the bits per pixel; *ck* is the hex value of the colorkey; *a* is the alpha value (0-255, inclusive); and *chan* is a concatenation of ‘R’ when there is a red mask, ‘G’ when there is a green mask, ‘B’ when there is a blue mask, and ‘A’ when there is an alpha mask.

init sel

Initialize SDL and the subsystems/configuration represented by sel (see [init flags], page 45).

init-subsystem sel

Initialize the SDL subsystems represented by sel. sel is a list of flags (symbols) from the same set useful for init.

quit

Shut down all SDL subsystems. Return #t.

quit-subsystem sel

Shut down the SDL subsystems represented by sel. sel is a list of flags (symbols) from the same set useful for init. Return #t.

was-init sel

Check if the SDL subsystems represented by sel have been initialized. sel is a list of flags (symbols) from the same set useful for init. Return a list likewise composed.

get-ticks

Return the number of milliseconds since the SDL library initialization.

delay ms

Wait ms milliseconds.

get-error

Return the current SDL error string.
3 Video

create-cursor data mask w h x y  [Procedure]
Return a new cursor from data and mask (both u8 uniform vectors), sized w by h and with hot pixel located at x,y.

create-yuv-overlay width height format [display]  [Procedure]
Create a new YUV overlay, sized width by height with overlay format (a symbol or an exact number). Optional arg display specifies a surface to use instead of creating a new one.

get-video-surface  [Procedure]
Return the current display surface.

video-cmf  [Procedure]
Return information about the video hardware as three values: capabilities (list of symbols), memory (integer), and format (pixel format object). The capabilities are:

 hw-available
wm-available
blit-hw  blit-hw-CC  blit-hw-A
blit-sw  blit-sw-CC  blit-sw-A
blit-fill

video-driver-name  [Procedure]
Return the name of the video driver.

list-modes [format [flags]]  [Procedure]
Return a list of available screen dimensions for pixel format and flags (see [video flags], page 47). Format defaults to that for the current screen. Flags default to none. Return #f if no modes are available, #t if all are available.

video-mode-ok width height bpp [flags]  [Procedure]
Check to see if a particular video mode is supported. Args are width, height, bpp (numbers), and flags (see [video flags], page 47). Return #f if the mode is not supported, or a number indicating the bits-per-pixel of the closest available mode supporting width and height.

set-video-mode width height bpp [flags]  [Procedure]
Set the SDL video mode with width, height and bits-per-pixel bpp. Optional arg flags (see [video flags], page 47) is supported. Return a new surface.

3.1 Rectangles

The external representation of a Rectangle object is:

#<SDL-Rect w x h y s>
where w and h are the width and height of the rectangle, and x and y are its horizontal and vertical coordinates. s may be ‘+’ or ‘-’.

rect? obj  [Procedure]
Return #t iff obj is an SDL-rectangle object.

make-rect x y width height  [Procedure]
Return a rectangle object with location x,y and dimensions width by height.

rect:x rect  [Procedure]
Get x from rect.
rect:y rect
Get y from rect.

rect:w rect
Get w from rect.

rect:h rect
Get h from rect.

rect:set-x! rect value
Set x in rect to value.

rect:set-y! rect value
Set y in rect to value.

rect:set-w! rect value
Set w in rect to value.

rect:set-h! rect value
Set h in rect to value.

update-rect surface x [y [w [h]]]
Update surface within a specified rectangle. The second arg can either be an SDL-Rect object, or the second through fifth args are numbers specifying the x, y, width and height of a rectangular area.

update-rects surface ls
On surface, update the rectangles in ls, a list of rectangles.

flip [surface]
Swap double buffers of the default surface, or of surface if specified.

3.2 Colors
The external representation of a Color object is:
#<SDL-Color r g b>
where r is the decimal value of the color object’s red component, and g and b the likewise respective green and blue components.

color? obj
Return #t iff obj is an SDL-Color object.

make-color r g b
Return a color object with r, g, and b components.

color:r color
Get r from color.

color:g color
Get g from color.

color:b color
Get b from color.

color:set-r! color value
Set r in color to value.
**Chapter 3: Video**

**color:set-g!**  
Set g in color to value.  

**color:set-b!**  
Set b in color to value.  

**set-colors!**  
Set a portion of the colormap for the 8-bit surface using colors, a vector of SDL-Colors. Optional arg start (an integer in the range [0,255]) specifies the portion to be modified. It defaults to 0.  

**set-palette**  
Set the palette of an 8-bit surface using flags (see [palette flags], page 47) and colors, a vector of SDL-Colors. Optional arg start (an integer in the range [0,255]) specifies the portion to be modified. It defaults to 0.  

**set-gamma**  
Set the color gamma function for the display using real numbers redgamma, greengamma and bluegamma.  

**get-gamma-ramp**  
Return the gamma translation lookup tables currently used by the display as a list of three tables, for red, green and blue. Each table is a u16 uniform vector of length 256. Return #f if unsuccessful.  

**set-gamma-ramp**  
Set the gamma translation lookup tables currently used by the display to tables r, g and b, each a u16 uniform vector of length 256, or #f, in which case that particular component is unchanged. Return #t if successful.  

**map-rgb**  
Map a RGB color value to the pixel format. The second arg can be an SDL-Color, otherwise the second through fourth args are red, green and blue values (numbers). Return the mapped components as an unsigned integer.  

**map-rgba**  
Map a RGB color value to the pixel format. If the second arg is an SDL-Color, the third is an alpha value (number). Otherwise, the second through fifth args are red, green, blue and alpha values (numbers). Return the mapped components as an unsigned integer.  

**pixel-rgb**  
Return RGB info from pixel in the specified pixel format as three values: r, g and b (all integers).  

**pixel-rgba**  
Return RGBA info from pixel in the specified pixel format as four values: r, g, b and a (all integers).  

**fill-rect**  
Fill surface rect with color (a number). If rect is #f, fill the entire surface. Return #t if successful.  

**display-format**  
Return a new surface made by converting surface to the display format. Return #f if not successful.
display-format-alpha surface
   Return a new surface made by converting surface to the display format, with an alpha channel.
   Return #f if not successful.

warp-mouse x y
   Set the position of the mouse cursor to x,y.

set-cursor cursor
   Set the current mouse cursor to cursor.

get-cursor
   Get the current mouse cursor.

show-cursor [setting]
   Return the current visibility of the pointer (aka “mouse cursor”) as a boolean. If arg setting
   (a boolean) is specified, set the visibility to setting (the returned visibility corresponds to
   that before the call, regardless).

gl-get-attribute attribute
   Return the value of a special SDL/OpenGL attribute.

gl-set-attribute attribute value
   Set the special SDL/OpenGL attribute to value. Both args are numbers.

lock-yuv-overlay overlay
   Lock the given YUV overlay. Return #f if successful.

unlock-yuv-overlay overlay
   Unlock the previously locked YUV overlay.

display-yuv-overlay overlay dstrect
   Blit the YUV overlay to the display dstrect over which it was created. Return #t if successful.

3.3 Windowing System Interaction

capTION title [icon]
   Set the title-bar and icon name of the display window to title and icon (both strings), re-
spectively. If icon is not specified, use title by default.

caption-ti
   Return display-window caption as two values: title and icon (both strings, or #f if not
   set).

set-icon icon
   Set icon for the display window.
**iconify-window**  
Iconify/Minimize the window. Return #t if successful.

**toggle-full-screen [surface]**  
Toggle the default video surface between windowed and fullscreen mode, if supported. Optional arg surface specifies another surface to toggle. Return #t if successful.

**grab-input [mode]**  
Grab mouse and keyboard input. Return new grab state. Optional arg mode (a symbol) specifies the kind of grab, one of query (the default), off or on.

**get-app-state**  
Return the current state of the application, a list of symbols. The list may include: ‘mouse-focus’, ‘inputfocus’, ‘active’.

### 3.4 Surface

The external representation of a Surface object is:

```lisp
;; normally:
#<SDL-Surface w h depth bpp lock>

;; when the object is not associated with a SDL_Surface:
#<SDL-Surface NULL>
```

where w and h are the width and height of the surface, and depth is its bit-depth (e.g., ‘32’ for an RGBA surface). If the surface is locked, lock displays as ‘L’, otherwise nothing.

**make-surface width height [flags]**  
Return a new surface of dimensions width by height. Optional third arg flags (see [video flags], page 47) further specifies the surface. Color depth and masks are those for the current video surface.

**create-rgb-surface flags width height depth rmask gmask bmask amask**  
Return an empty surface. The eight arguments, directly analagous to those for SDL_CreateRGBSurface, are: flags (list of symbols, see [video flags], page 47), width, height, depth, rmask, gmask, bmask, amask (all numbers).

**surface:w surface**  
Get w from surface.

**surface:h surface**  
Get h from surface.

**surface:depth surface**  
Get format->BitsPerPixel from surface.

**surface:flags surface**  
Return flags from surface as a (possibly empty) list of symbols.

**surface-get-format surface**  
Return a new pixel format, the same used by surface.

**surface? obj**  
Return true iff obj is a surface.
lock-surface surface
Lock surface for direct access. Return #t if successful.

unlock-surface surface
Unlock previously locked surface.

load-bmp filename
Load bitmap data from filename. Return a new surface if successful, otherwise #f.

load-image filename
Load image data from filename. Return a new surface if successful, otherwise #f.

save-bmp surface filename
Save surface to filename in Windows BMP format. Return #t if successful.

surface-color-key! surface pixel [rle]
Set the color key for surface to pixel. If pixel is #f, clear the current color key. Otherwise, it should be an integer of the appropriate depth for surface (e.g., in the range \([0,65535]\) for 16 bpp). If color key processing is enabled, optional arg rle is a boolean that enables (true) or disables (false, the default) RLE acceleration. Return #t if successful.

surface-alpha! surface alpha [rle]
Set alpha blending for the entire surface to alpha. If alpha is #f, disable alpha blending. Otherwise it should be an integer in the range \([0,255]\) or one of the symbols transparent or opaque. Alpha blending is enabled, optional arg rle is a boolean that enables (true) or disables (false, the default) RLE acceleration. Return #t if successful.

set-clip-rect! surface [rect]
Set surface clipping rectangle to the whole surface. Optional arg rect, if non-#f, specifies a particular rectangle instead of using the whole surface.

get-clip-rect surface
Return the clipping rectangle for surface.

convert-surface surface format [flags]
Convert surface to the same format as another surface. Optional third arg flags is a list of flags (see [video flags], page 47).

blit-surface src [srcrect [dst [dstrect]]]
Perform a fast blit from the src surface srcrect to the dst surface dstrect. srcrect defaults to x=0, y=0, src surface dimensions. If unspecified dst is taken as the default video surface. dstrect likewise defaults to x=0, y=0, dst surface dimensions.

3.5 Misc Surface Operations

vertical-flip-surface surface
Return a new surface created by flipping surface vertically.

horizontal-flip-surface surface
Return a new surface created by flipping surface horizontally.

vh-flip-surface surface
Return a new surface created by flipping surface both vertically and horizontally.


**surface-pixels surface [squash]**  
[Procedure]  
Return pixel data of `surface` as a new uniform vector. The uvec has type `u8`, `u16` or `u32`, corresponding to the `surface` depth, with `height` x `width` elements. A 24bpp surface — `depth-in-bytes` of 3 — is expanded (per pixel) to `u32`, leaving the high byte clear.

Optional arg `squash` non-#f means to return a u8vector regardless of `surface` depth, with `height` x `width` x `depth-in-bytes` elements.

**must-lock? surface**  
[Procedure]  
Return #t if `surface` needs to be locked before access. Failure to do so may result in a segfault.
4 Events

make-event [type]  [Procedure]
Return a new SDL event. Optional arg type is a symbol (see [event-type enums], page 45). If omitted, the default is SDL_NOEVENT.

event:type event  [Procedure]
Return the symbolic type from event.

event:set-type! event value  [Procedure]
Set type in event to value, a symbol or integer.

4.1 Activity

The value for event:active:gain and event:active:set-gain! is a symbol, one of: gained or lost.
The value for event:active:state and event:active:set-state! is a (possibly empty) list of symbols from the same set used by get-app-state.

event:active:gain event  [Procedure]
Return the symbolic active.gain from event.

event:active:state event  [Procedure]
Return active.state from event as a (possibly empty) list of symbols.

event:active:set-gain! event value  [Procedure]
Set active.gain in event to value, a symbol or integer.

event:active:set-state! event value  [Procedure]
Set active.state in event to value, a (possibly empty) list of symbols.

4.2 Keys

The value for event:key:state and event:key:set-state! is a symbol, one of: released or pressed.

event:key:keysym:sym event  [Procedure]
Return the symbolic key.keysym.sym from event.

event:key:keysym:set-sym! event value  [Procedure]
Set key.keysym.sym in event to value, a symbol or integer.

event:key:keysym:mod event  [Procedure]
Return key.keysym.mod from event as a (possibly empty) list of symbols.

event:key:keysym:set-mod! event value  [Procedure]
Set key.keysym.mod in event to value, a (possibly empty) list of symbols.

event:key:state event  [Procedure]
Return the symbolic key.state from event.

event:key:keysym:scancode event  [Procedure]
Get key.keysym.scancode from event.

event:key:keysym:unicode event  [Procedure]
Get key.keysym.unicode from event.
**event:key:set-state!** event value
Set key.state in event to value, a symbol or integer.

**event:key:keysym:set-scancode!** event value
Set key.keysym.scancode in event to value.

**event:key:keysym:set-unicode!** event value
Set key.keysym.unicode in event to value.

### 4.3 Motions

**event:motion:state** event
Return motion.state from event as a (possibly empty) list of symbols.

**event:motion:x** event
Get motion.x from event.

**event:motion:y** event
Get motion.y from event.

**event:motion:xrel** event
Get motion.xrel from event.

**event:motion:yrel** event
Get motion.yrel from event.

**event:motion:set-state!** event value
Set motion.state in event to value, a (possibly empty) list of symbols.

**event:motion:set-x!** event value
Set motion.x in event to value.

**event:motion:set-y!** event value
Set motion.y in event to value.

**event:motion:set-xrel!** event value
Set motion.xrel in event to value.

**event:motion:set-yrel!** event value
Set motion.yrel in event to value.

### 4.4 Buttons

The value for **event:button:button** and **event:button:set-button!** is a (possibly empty) list of symbols from the set:

- left
- middle
- right
- wheel-up
- wheel-down
- x1
- x2

The value for **event:button:state** and **event:button:set-state!** is a symbol, one of: released or pressed.

**event:button:button** event
Return the symbolic button.button from event.

**event:button:state** event
Return the symbolic button.state from event.
event:button:x event
Get button.x from event.  [Procedure]

event:button:y event
Get button.y from event.  [Procedure]

event:button:set-button! event value
Set button.button in event to value, a symbol or integer.  [Procedure]

event:button:set-state! event value
Set button.state in event to value, a symbol or integer.  [Procedure]

event:button:set-x! event value
Set button.x in event to value.  [Procedure]

event:button:set-y! event value
Set button.y in event to value.  [Procedure]

4.5 Joysticks
The value for event:jbutton:state and event:jbutton:set-state! is a symbol, one of: released or pressed.
The value for event:jhat:value and event:jhat:set-value! is a list of or more symbols from the set:

centered
up down
left right

Specifying the empty list for event:jhat:set-value! is effectively the same as specifying centered.

event:jaxis:which event
Get jaxis.which from event.  [Procedure]

event:jaxis:axis event
Get jaxis.axis from event.  [Procedure]

event:jaxis:value event
Get jaxis.value from event.  [Procedure]

event:jaxis:set-which! event value
Set jaxis.which in event to value.  [Procedure]

event:jaxis:set-axis! event value
Set jaxis.axis in event to value.  [Procedure]

event:jaxis:set-value! event value
Set jaxis.value in event to value.  [Procedure]

event:jbutton:which event
Get jbutton.which from event.  [Procedure]

event:jbutton:button event
Get jbutton.button from event.  [Procedure]

event:jbutton:state event
Return the symbolic jbutton.state from event.  [Procedure]
event:jbutton:set-which! event value
  Set jbutton.which in event to value.

event:jbutton:set-button! event value
  Set jbutton.button in event to value.

event:jbutton:set-state! event value
  Set jbutton.state in event to value, a symbol or integer.

event:jball:which event
  Get jball.which from event.

event:jball:ball event
  Get jball.ball from event.

event:jball:xrel event
  Get jball.xrel from event.

event:jball:yrel event
  Get jball.yrel from event.

event:jball:set-which! event value
  Set jball.which in event to value.

event:jball:set-ball! event value
  Set jball.ball in event to value.

event:jball:set-xrel! event value
  Set jball.xrel in event to value.

event:jball:set-yrel! event value
  Set jball.yrel in event to value.

event:jhat:which event
  Get jhat.which from event.

event:jhat:hat event
  Get jhat.hat from event.

event:jhat:value event
  Return jhat.value from event as a (possibly empty) list of symbols.

event:jhat:set-which! event value
  Set jhat.which in event to value.

event:jhat:set-hat! event value
  Set jhat.hat in event to value.

event:jhat:set-value! event value
  Set jhat.value in event to value, a (possibly empty) list of symbols.
4.6 Resizes

**event:resize:w event**
Get resize.w from event.

**event:resize:h event**
Get resize.h from event.

**event:resize:set-w! event value**
Set resize.w in event to value.

**event:resize:set-h! event value**
Set resize.h in event to value.

4.7 Misc

**pump-events**
Gather events from input devices and update the event queue.

**evqueue-add [events...]**
Add events to the back of the event queue. Return the count of successfully added events.

**evqueue-peek n mask [accumulate]**
Return a count (less than or equal to n) of events at the front of the event queue that match mask, without changing the queue. Optional arg accumulate if non-#f means to return the list of matched events, instead. If there are errors, return #f.

See [event-mask flags], page 44.

**evqueue-get n mask**
Return a list (of length at most n) of events at the front of the event queue that match mask, removing them from the queue. If there are errors, return #f.

See [event-mask flags], page 44.

**poll-event [event]**
Poll for events and return #t if there are any pending. Optional arg event specifies an event object (from make-event) to be filled in with the next event from the queue (if available).

**wait-event [event]**
Wait indefinitely for and return #f only if there were errors. Optional arg event specifies an event object (from make-event) to be filled in with the next event from the queue.

**push-event event**
Push event onto the queue. Return #t on success.

**set-event-filter filter full?**
Set the event filter to filter, or clear it if filter is #f. This is a procedure called with one arg, and whose return value, if non-#f, means to keep the event, otherwise discard it. If full? is #f, the arg the event type (a symbol), otherwise it is an event object.

**get-event-filter**
Return information on the current event filter, or #f if none is set. If there is a filter, the value is a pair with car the filter proc, and cdr #f if the proc takes an event type, or #t if the proc takes an event object.
event-type-handling type [setting]  

Return \#t if event type (see [event-type enums], page 45) is recognized and queued, or \#f if it is ignored. If setting is specified, set the handling of type to the truth value of setting first.

able-unicode [enable-p]  

Return \#t iff UNICODE keyboard translation is enabled. Optional arg enable? if non-\#f, enables UNICODE keyboard translation, or disables it if \#f.

enable-key-repeat delay interval  

Enable or disable keyboard repeat. delay is the initial delay in ms between the time when a key is pressed, and keyboard repeat begins. interval is the time in ms between keyboard repeat events. If delay is 0, keyboard repeat is disabled. Return \#t on success.

get-key-state  

Return a list of pressed keys (see [keysym enums], page 45).

get-mod-state  

Return the current key modifier state as a list of symbols.

set-mod-state modstate  

Set the current key modifier state to modstate, a list of symbols. This does not change the keyboard state, only the key modifier flags.

button? mask  

Return \#t if buttons specified in mask are pressed, otherwise \#f. mask is a symbol or a list of symbols from the set returned by get-mouse-state.

For backward compatibility, mask can also be the (integer) logior of the buttons, using mapping:

1 left
2 middle
4 right
8 wheel-up
16 wheel-down
32 x1
64 x2

For example, a value of 5 specifies both left and right buttons, equivalent to (left right).

mouse-bxy [relative]  

Return three values: a (possibly empty) list of symbols representing pressed mouse buttons (like event:button:button), and two integer coordinates x and y.

Optional arg relative non-\#f means the coordinates are relative to the last time the underlying SDL_GetRelativeMouseState was called.
5 Joystick

The external representation of a Joystick object is:

`#<SDL-Joystick index>`

where `index` is a decimal number of the joystick, or ‘-1’ if the object is not associated with a joystick.

**num-joysticks**
Return the number of joysticks.

**joystick? obj**
Return `#t` iff `obj` is a joystick object.

**joystick-name [n]**
Return the (string) name of the default joystick, or `#f`. Optional arg `n` specifies which joystick to check.

**joystick-open [n]**
Return a handle to the default joystick opened for use. Optional arg `n` specifies which joystick to open.

**joystick-opened? [n]**
Return `#t` iff the default joystick is opened. Optional arg `n` specifies which joystick to check.

**joystick-index joystick**
Return the index of `joystick`.

**joystick-num-axes joystick**
Return the number of axes for `joystick`.

**joystick-num-balls joystick**
Return the number trackballs for `joystick`.

**joystick-num-hats joystick**
Return the number of hats for `joystick`.

**joystick-num-buttons joystick**
Return number of buttons for `joystick`.

**joystick-update**
Update the state of all Joysticks.

**joystick-polling [setting]**
Return `#t` if joystick events are polled and queued (such that it is unnecessary to “manually” call `joystick-update`), otherwise `#f`. If `setting` is specified, set joystick events polling to the truth value of `setting` first.

**joystick-get-axis joystick axis**
For joystick, return state of `axis`.

**joystick-ball-xy joystick n**
Return relative motion of `joystick` trackball `n` as two values: dx and dy (both integers).

**joystick-get-hat joystick n**
For joystick, return state of hat `n`. 
joystick-get-button joystick n
   For joystick, return state of button n, a symbol, one of: released or pressed.

joystick-close joystick
   Close a previously opened joystick.
The external representation of a CDROM Drive object is:

```
#<SDL-CD [status]>
```

where `status` is one of ‘TRAY EMPTY’, ‘STOPPED’, ‘PLAYING’, ‘PAUSED’, ‘DRIVE ERROR’, or ‘???’.
(Normally, the last one should never appear.)

### Procedures

- **cd? obj**
  Return `#t` iff `obj` is a CDROM drive object.

- **cd-num-drives**
  Return the number of CDROM drives.

- **cd-name [drive]**
  Return a human-readable, system-dependent identifier (a string) for the CDROM, or `#f`.
  Optional arg `drive` is a number specifying which drive.

- **cd-open [drive]**
  Open the CDROM drive for access and return its handle. If the drive is unavailable, return `#f`.
  Optional arg `drive` is a number specifying which drive.

- **cd-status cdrom**
  Return the current status of the drive `cdrom` as a symbol (see [cdrom-state enums], page 44).

- **cd-in-drive? cdrom**
  Return `#t` iff there is a CD in drive `cdrom`.

- **cd-get-num-tracks cdrom**
  Return the number of tracks on the CD in drive `cdrom`.

- **cd-get-cur-track cdrom**
  Return the current track on the CD in drive `cdrom`.

- **cd-get-cur-frame cdrom**
  Return the current frame of the CD in drive `cdrom`.

- **cd-nth-track-itlo cdrom [n]**
  For CD in drive `cdrom`, return four values describing track `n` (zero if unspecified): `id, type, length` and `offset`, all integers except for `type`, which is a symbol, either `audio` or `data`.

- **cd-play-tracks cdrom [start-track [start-frame [n-tracks [n-frames]]]]**
  Play the given CD tracks in drive `cdrom`. Play the CD starting at `start-track` and `start-frame` for `ntracks` tracks and `nframes` frames. If both `ntrack` and `nframe` are 0, play until the end of the CD. This procedure will skip data tracks, and should only be called after calling `cd-status` to get track information about the CD. Return `#t` if successful.

- **cd-play cdrom start length**
  Play CD in drive `cdrom` from `start` frame for `length` frames. Return `#t` if successful.

- **cd-pause cdrom**
  Pause the CD in drive `cdrom`. Return `#t` if successful.

- **cd-resume cdrom**
  Resume (unpause) the CD in drive `cdrom`. Return `#t` if successful.

- **cd-stop cdrom**
  Stop the CD in drive `cdrom`. Return `#t` if successful.
cd-eject cdrom
Eject the CD from drive cdrom. Return #t if successful.

[Procedure]

cd-close cdrom
Close the drive cdrom.

[Procedure]

cd-msf->frames m [s [f]]
Return frames (an integer) computed fr m, second s and frame f. s and f are optional.

[Procedure]

frames-msf frames
Break down frames (an integer) and return three values: minute, second and frames (all integers).
7 OpenGL

todo
8 TrueType

**ttf-init** [Procedure]
Initialize the SDL_ttf subsystem.

**load-font** file ptsize [Procedure]
Load a font from file with point size ptsize. Return a handle.

**font:style** font [Procedure]
Return the style of font (see [font-style flags], page 45). This font style is implemented by modifying the font glyphs, and doesn’t reflect any inherent properties of the truetype font file.

**font:set-style!** font style [Procedure]
Set font style to style (see [font-style flags], page 45). This font style is implemented by modifying the font glyphs, and doesn’t reflect any inherent properties of the truetype font file.

**font:height** font [Procedure]
Return the total height of font, usually equal to point size.

**font:ascent** font [Procedure]
Return the offset from the baseline to the top of font. This is a positive number.

**font:descent** font [Procedure]
Return the offset from the baseline to the bottom of font. This is a negative number.

**font:line-skip** font [Procedure]
Return the recommended spacing between lines of text for font.

**font:glyph-xXyYa** font ch [Procedure]
Return the metrics (dimensions) of a glyph as five values. The glyph is a font-specific rendering of char ch. Values are: minx, maxx, miny, maxy and advance (all integers).

**text-wh** font text [Procedure]
Return two values: width and height (both integers) representing the dimensions of the font-specific rendering of the string text.

**utf8-wh** font text [Procedure]
Return two values: width and height (both integers) representing the dimensions of the font-specific rendering of the UTF-8 string text.

**render-text** font text fg [bg] [Procedure]
Return a new surface containing the font-specific rendering of the text string. Third argument is the foreground color; optional fourth argument is the background color, or #t if the text is to be blended.

**render-utf8** font text fg [bg] [Procedure]
Return a new surface containing a font-specific rendering of the utf8 string text. Third argument is the foreground color; optional fourth argument is the background color, or #t if the text is to be blended.

**render-glyph** font ch fg [bg] [Procedure]
Return a new surface containing a font-specific rendering of the character ch. Third argument is the foreground color; optional fourth argument is the background color, or #t if the text is to be blended.

**ttf-quit** [Procedure]
Quit the SDL_ttf subsystem.
9 Audio

open-audio [freq [format [stereo [chunksize]]]]

Open the mixer with a certain audio format. Optional args freq (number), format (number), stereo (boolean) and chunksize (number) specify those aspects of the device. Return #t if successful.

allocated-channels numchans

Dynamically change the number of channels managed by the mixer to numchans. If decreasing the number of channels, the upper channels are stopped. Return the new number of allocated channels.

device-ffc

Return audio device parameters as three values: frequency (Hz), format (number of bits) and channels (number of allocated channels).

load-music filename

Load music data (.mod .s3m .it .xm) from filename. Return a new music object if successful, otherwise #f.

load-wave filename

Load wave data from filename. Return a new audio object if successful, otherwise #f.

reserve-channels num

Reserve the first num channels (0 through num-1) for the application. In other words don’t allocate them dynamically to the next sample if requested with a -1 value below. Return the number of reserved channels.

group-channel channel [tag]

Attach to channel a tag. A tag can be assigned to several mixer channels, to form groups of channels. If tag is not specified, or is -1, the tag is removed (actually -1 is the tag used to represent the group of all the channels). Return #t if successful.

group-channels from to [tag]

Assign channels in the range from through to to the default group. Optional arg tag specifies the group to use. Return #t if successful.

group-available [tag]

Return the first available channel in the default group of channels. Optional arg tag specifies the group to check.

group-count [tag]

Return the number of channels in the default group. Optional arg tag specifies the group to check.

group-oldest [tag]

Return the "oldest" sample playing in the default group of channels. Optional arg tag specifies the group to check.

group-newer [tag]

Return the "most recent" (i.e. last) sample playing in the default group of channels. Optional arg tag specifies the group to check.
play-channel chunk [channel [loops [ticks [fade]]]]

Play an audio chunk on a specific channel. If the channel is unspecified or is -1, play on the first free channel. If loops is specified and greater than zero, loop the sound that many times. If loops is -1, loop infinitely (~65000 times). If ticks is specified, stop after that number of ticks. If fade is specified, fade in over that number of milliseconds. Return which channel was used to play the sound.

play-music music [loops [fade]]

Play a music track. Optional args loops and fade are as in play-channel.

volume [volume [which]]

Return the current volume on the default channel. Optional arg volume (a number in the range 0-128) means set the volume to volume and return the original volume. Optional second arg which specifies a chunk or channel to check (or modify) instead of the default. If volume is non-#f and which is #f, modify all channels.

[Here is the original (perhaps clearer) docstring. —ttn]

Set the volume in the range of 0-128 of a specific channel or chunk. If the channel is unspecified or is -1, set volume for all channels. Return the original volume. If the volume is unspecified or is -1, just return the current volume.

music-volume [volume]

Return the current volume. Optional arg volume (a number in the range 0-128) means set the volume to volume.

halt-channel [channel]

Halt playing of the default channel. Optional arg channel specifies a channel to halt.

halt-group [tag]

Halt playing of the default group. Optional arg tag specifies the group to halt.

halt-music

Halt playing of the music.

expire-channel [channel [ticks]]

Turn off expiration for the default channel. Optional arg channel specifies a channel to change. Optional arg ticks (a number) means set the expiration delay to that many milliseconds, rather than turning it off.

fade-out-channel [which [ms]]

Halt a channel, fading it out progressively until silent. Optional arg which specifies a channel to halt. Second optional arg ms specifies the number of milliseconds the fading will take (default 0).

fade-out-group [tag [ms]]

Halt a group, fading it out progressively until silent. Optional arg tag specifies a group to halt. Second optional arg ms specifies the number of milliseconds the fading will take (default 0).

fade-out-music [ms]

Halt the music, fading it out progressively until silent. Optional arg ms specifies the number of milliseconds the fading will take (default 0).

fading-music

Return the fading status of the music, one of the symbols: no, out, in.
fading-channel [which]
Return the fading status (a symbol, see fading-music) of the default channel. Optional arg
which selects which channel to check.

pause [channel]
Pause the default channel. Optional arg channel selects which channel to pause.

resume [channel]
Resume (unpause) the default channel. Optional arg channel selects which channel to resume.

paused? [channel]
Return #t if the default channel is paused. Optional arg channel selects a which channel to
check.

pause-music
Pause the music.

resume-music
Resume (unpause) the music.

rewind-music
Rewind the music.

paused-music?
Return #t if the music is currently paused.

playing? [channel]
Return #t iff the default channel is playing. Optional arg channel selects which channel to
check.

playing-music?
Return #t iff the music is currently playing.

set-music-command command
Stop music and set external music playback command to command, a string. As a special
case, if command is #f, arrange to use internal playback, instead.

FWIW, the C header file for the following panning, distance and position procs says:
Setting (channel) to MIX_CHANNEL_POST registers this as a posteffect, and the
panning will be done to the final mixed stream before passing it on to the audio
device.

set-panning channel l r
Set panning for (stereo) channel with l and r. Both l and r are integers 0–255, inclusive,
where 0 is quietest and 255 is loudest.
To get “true” panning, use (set-panning CH N (- 255 N)).

set-distance channel distance
Set the “distance” of channel to distance (integer, 0–255). This controls the location of the
sound with respect to the listener.
Distance 0 is overlapping the listener, and 255 is as far away as possible. A distance of
255 does not guarantee silence; in such a case, you might want to try changing the chunk’s
volume, or just cull the sample from the mixing process with halt-channel.
For efficiency, the precision of this effect may be limited (distances 1 through 7 might all
produce the same effect, 8 through 15 are equal, etc).
Setting (distance) to 0 unregisters this effect, since the data would be unchanged.
**set-position channel angle distance**  
Set the “position” of channel to angle, distance. In this polar coordinate, angle is in degrees (integer modulo 360), and distance is an integer 0–255 (and is treated as in proc *set-distance* – see notes there).

Angle 0 is due north, and rotates clockwise as the value increases. For efficiency, the precision of this effect may be limited (angles 1 through 7 might all produce the same effect, 8 through 15 are equal, etc).

Setting angle and distance to 0 unregisters this effect, since the data would be unchanged.

Additionally, the C header says:

If the audio device is configured for mono output, then you won’t get any effectiveness from the angle; however, distance attenuation on the channel will still occur. While this effect will function with stereo voices, it makes more sense to use voices with only one channel of sound, so when they are mixed through this effect, the positioning will sound correct. You can convert them to mono through SDL before giving them to the mixer in the first place if you like.

**close-audio**  
Close the mixer, halting all playing audio.
10 SDL gfx by Andreas Schiffler

10.1 Graphics Primitives

**draw-point** surface x y color
- On surface, draw a point at location x,y with color color.

**draw-hline** surface x1 x2 y color
- On surface, draw a horizontal line segment from x1,y to x2,y, with color color.

**draw-vline** surface x y1 y2 color
- On surface, draw a vertical line segment from x,y1 to x,y2, with color color.

**draw-rectangle** surface x1 y1 x2 y2 color
- On surface, draw a rectangle with opposite points x1,y1 and x2,y2, with color color. Optional arg fill means to fill the rectangle as well.

**draw-rounded-rectangle** surface x1 y1 x2 y2 rad color
- On surface, draw a rectangle with opposite points x1,y1 and x2,y2, with rounded corners radius rad in color color. Optional arg fill means to fill the rectangle as well.

**draw-line** surface x1 y1 x2 y2 color
- On surface, draw a line segment from x1,y1 to x2,y2, with color color.

**draw-aa-line** surface x1 y1 x2 y2 color
- On surface, draw an anti-aliased line segment from x1,y1 to x2,y2, with color color.

**draw-thick-line** surface x1 y1 x2 y2 width color
- On surface, draw a line segment from x1,y1 to x2,y2, with thickness width in color color.

**draw-arc** surface x y r start end color
- On surface, draw arc with center x,y and radius r, going from start to end (degrees), with color color. If start is greater than end, the effective range of the arc is taken to be end to start (that is, these arguments are internally reversed).

**draw-circle** surface x y r color
- On surface, draw a circle with center x,y and radius r, with color color. Optional arg fill means to fill the circle as well.

**draw-aa-circle** surface x y r color
- On surface, draw an anti-aliased circle with center x,y and radius r, with color color.

**draw-ellipse** surface x y rx ry color
- On surface, draw an ellipse with center x,y x-radius rx, y-radius ry, with color color. Optional arg fill means to fill the ellipse as well.

**draw-aa-ellipse** surface x y rx ry color
- On surface, draw an anti-aliased ellipse with center x,y, x-radius rx, y-radius ry, with color color.

**draw-pie-slice** surface x y rad start end color
- On surface, draw a pie slice with center x,y and radius rad, going from start to end (degrees), with color color. Optional arg fill means to fill the slice as well.
**draw-trigon**  
`surface x1 y1 x2 y2 x3 y3 color [fill]`  
On `surface`, draw a triangle with vertices at `x1,y1`, `x2,y2` and `x3,y3`, with color `color`. Optional arg `fill` means to fill the triangle as well.

**draw-aa-trigon**  
`surface x1 y1 x2 y2 x3 y3 color`  
On `surface`, draw an anti-aliased triangle with vertices at `x1,y1`, `x2,y2` and `x3,y3`, with color `color`.

**draw-polygon**  
`surface vx vy color [fill]`  
On `surface`, draw a polygon whose points are specified by corresponding pairs from the s16 uniform vectors `vx` and `vy`, in color `color`. Optional arg `fill` means to fill the polygon as well.

**draw-aa-polygon**  
`surface vx vy color`  
On `surface`, draw an anti-aliased polygon whose points are specified by corresponding pairs from the s16 uniform vectors `vx` and `vy`, in color `color`.

**draw-textured-polygon**  
`surface vx vy texture tdx tdy`  
On `surface`, draw a polygon whose points are specified by corresponding pairs from the s16 uniform vectors `vx` and `vy`, filling from `texture` (a surface) with offset `tdx`, `tdy`.

**draw-bezier**  
`surface vx vy s color`  
On `surface`, draw a bezier curve whose points are specified by corresponding pairs from the s16 uniform vectors `vx` and `vy`, with `s` steps in color `color`.

**draw-character**  
`surface x y c color`  
On `surface` at position `x,y`, draw char `c` with color (a number).

**draw-string**  
`surface x y text color`  
On `surface` at position `x,y`, draw string `text` with color (a number).

**font-rotation!**  
`rotation`  
Set the rotation for glyphs drawn by `draw-character` and `draw-string` to `rotation` (an integer or symbol), one of:  
0 none  
1 clockwise  
2 upside-down  
3 counter-clockwise

### 10.2 Rotation / Zooming

For `roto-zoom-surface` and `roto-zoom-surface-xy`, a positive angle means counter-clockwise rotation.

**roto-zoom-surface**  
`surface angle [zoom [smooth]]`  
Return a new surface made from rotating `surface` by `angle` degrees. Optional third arg `zoom` (default value 1.0) changes the size as well. Optional fourth arg `smooth` turns on anti-aliasing.

**roto-zoom-surface-xy**  
`surface angle [zoomx [zoomy [smooth]]]`  
Return a new surface made from rotating `surface` by `angle` degrees. Optional third and fourth args `zoomx` and `zoomy` (default value 1.0 for both) changes the size as well. Optional fifth arg `smooth` turns on anti-aliasing.

**zoom-surface**  
`surface zoomx [zoomy [smooth]]`  
Return a new scaled copy of `surface`. `zoomx` and `zoomy` specify the scaling factor. If omitted, `zoomy` defaults to `zoomx`. Optional fourth arg `smooth` turns on anti-aliasing.
shrink-surface surface factorx factory  
Return a new shrunken copy of surface. factorx and factory are positive integers specifying the inverse scaling factor. For example, 2 means half size, 3 means one-third size, etc.

The returned surface is antialiased by “averaging the source box RGBA or Y information” and is in 32-bit RGBA format.

10.3 Managing Frame Rate

The external representation of an FPS Manager object is:

```lisp
#<FPS-manager rHz>
```

where r is the decimal framerate that the object manages.

make-fps-manager [n]  
Return a FPS manager object to be passed as the first arg to fps-manager-set!, fps-manager-get and fps-manager-delay!. Optional arg n specifies the value in Hz to initialize the object (default 30 if not specified).

fps-manager-set! mgr n  
Arrange for FPS manager mgr to try to maintain a frame rate of n Hz. Return #f if not successful.

fps-manager-get mgr  
Return the frame rate of FPS manager mgr in Hz, or #f if unsuccessful.

fps-manager-count mgr  
Return the frame count of FPS manager mgr, or #f if unsuccessful. A frame is counted each time fps-manager-delay! is called.

fps-manager-delay! mgr  
Request an appropriate delay from FPS manager mgr. For some versions of SDL_gfx (not the embedded one, currently), return the number of milliseconds elapsed since the last call. This value may be 0 (zero).

10.4 RGBA Extras

set-pixel-alpha! surface alpha  
If surface is 32-bit, set each pixel’s alpha value to alpha, an integer 0-255, inclusive, and return #t. Otherwise, do nothing and return #f.

multiply-pixel-alpha! surface factor  
Multiply the alpha channel of 32-bit surface by factor, an integer 0-255, inclusive. The result is scaled back; the effective factor is factor/256. Return #t if alpha was changed.

blit-rgba src srect dst drect  
Blit from 32-bit surface src rectangle srect to 32-bit surface dst rectangle drect. Return #f if there were problems (use get-error for more info); 1 if a blit was performed; 0 otherwise.

Both srect and drect may be #f to indicate the entire surface as source or destination.
10.5 Image Filtering

The image filtering procedures take one or more surfaces — the "source(s)", identified by \( s, s_1, s_2 \) — and perform an operation on them, writing the result to the "destination" (\( d \)) surface. The sources and destination must all have the same width, height and pixel format. (This pixel format requirement may be relaxed in the future.) The procedures return \#t on success, else \#f.

With the exception of four procedures: \texttt{imfi-add-c}, \texttt{imfi-sub-c}, \texttt{imfi-lshr}, and \texttt{imfi-lshl}, all bytes in a pixel are subject to the same operation.

For \texttt{imfi-add-c} and \texttt{imfi-sub-c}, if the \( c \) (constant) argument value is more than 255 (does not fit in 8 bits), the constant is interpreted as a field of four channels and the operation occurs on a per-channel basis; otherwise (constant fits in 8 bits), a byte-wise operation is performed as usual.

For \texttt{imfi-lshr}, and \texttt{imfi-lshl}, the operation is done pixel-wise (on a logical pixel, assumed bit-depth 32). This is indicated by the ‘\texttt{(uint)}’ cast in their descriptions.

\texttt{imfi-mmx? [setting]} [Procedure]

If \texttt{setting} is \#t, enable MMX instructions for the image filter proc (if possible); if \#f, disable; otherwise do nothing. Return the (boolean) value of the setting afterwards.

\texttt{imfi-add src1 src2 dst} [Procedure]

\[ D = \text{saturation255} \ (S_1 + S_2). \]

\texttt{imfi-mean src1 src2 dst} [Procedure]

\[ D = S_1/2 + S_2/2. \]

\texttt{imfi-sub src1 src2 dst} [Procedure]

\[ D = \text{saturation0} \ (S_1 - S_2). \]

\texttt{imfi-abs-diff src1 src2 dst} [Procedure]

\[ D = |S_1 - S_2|. \]

\texttt{imfi-mult src1 src2 dst} [Procedure]

\[ D = \text{saturation} \ (S_1 * S_2). \]

\texttt{imfi-mulnor src1 src2 dst} [Procedure]

\[ D = S_1 * S_2 \ (\text{non-MMX}). \]

\texttt{imfi-muldiv2 src1 src2 dst} [Procedure]

\[ D = \text{saturation255} \ (S_1/2 * S_2). \]

\texttt{imfi-muldiv4 src1 src2 dst} [Procedure]

\[ D = \text{saturation255} \ (S_1/2 * S_2/2). \]

\texttt{imfi-logand src1 src2 dst} [Procedure]

\[ D = S_1 \& S_2. \]

\texttt{imfi-logior src1 src2 dst} [Procedure]

\[ D = S_1 | S_2. \]

\texttt{imfi-div src1 src2 dst} [Procedure]

\[ D = S_1 / S_2 \ (\text{non-MMX}). \]

\texttt{imfi-not src dst} [Procedure]

\[ D = !S. \]
### Chapter 10: SDL gfx by Andreas Schiffler

#### Procedures

**imfi-add-c** *src dst c*
\[
D = \text{saturation}255(S + C).
\]

**imfi-add-c-to-half** *src dst c*
\[
D = \text{saturation}255(S/2 + C).
\]

**imfi-sub-c** *src dst c*
\[
D = \text{saturation}0(S - C).
\]

**imfi-ashr** *src dst n*
\[
D = \text{saturation}0(S \gg N).
\]

**imfi-lshr** *src dst n*
\[
D = \text{saturation}0((\text{uint}) S \gg N).
\]

**imfi-mul-c** *src dst c*
\[
D = \text{saturation}255(S \times C).
\]

**imfi-ashr-mul-c** *src dst n c*
\[
D = \text{saturation}255((S \gg N) \times C).
\]

**imfi-bshl** *src dst n*
\[
D = (S \ll N).
\]

**imfi-lshl** *src dst n*
\[
D = ((\text{uint}) S \ll N).
\]

**imfi-ashl** *src dst n*
\[
D = \text{saturation}255(S \ll N).
\]

**imfi-binarize** *src dst t*
\[
D = (S < T ? 0 : 255).
\]

**imfi-clip** *src dst tmin tmax*
\[
D = (\text{Tmin} \leq S \leq \text{Tmax}) \, ? \, 255 : 0.
\]

**imfi-normalize-linear** *src dst cmin cmax nmin nmax*
\[
D = \text{saturation}255(((\text{Nmax} - \text{Nmin}) / (\text{Cmax} - \text{Cmin}) \times (S - \text{Cmin}) + \text{Nmin})).
\]
11 Miscellaneous Utilities

These are available in module (sdl misc-utils).

**exact-truncate number**  
Return the exact truncation (rounding to zero) of number. This is “safer” than simply inexact->exact for some Guile versions.

```scheme
(define scale 0.180281690140845)  
(inexact->exact scale)  
⇒ 3247666210160131/18014398509481984 ; Guile 1.8.7  
⇒ 0 ; Guile 1.4.x  
(exact-truncate scale)  
⇒ 0
```

**exact-floor number**  
Return the exact floor (rounding to negative infinity) of number.

**call-with-clip-rect rect thunk**  
Set default clip rect to rect, call thunk, and restore it. thunk is a procedure that takes no arguments.

**create-rgba-surface w h**  
Return a new 32bpp RGBA surface with dimensions w by h pixels. The surface has flag src-alpha set. The masks are as follows:

- red #x000000FF
- green #x0000FF00
- blue #x0FF0000
- alpha #xFF000000

**create-rgba-square edge-len**  
Return a new 32bpp RGBA square surface with edge-len sides. (Both width and height are edge-len, an integer.)

**rotate-square square angle [mksquare]**  
Return a new surface made by rotating square by angle degrees. The square retains its original size. If the new surface has flag src-alpha set, use (sdl gfx) blit-rgba, otherwise (sdl sdl) blit-surface, for the resizing blit.

Optional arg mksquare is a procedure of one arg that creates a square surface. If unspecified, use create-rgba-square.

See roto-zoom-surface for interpretation of angle.

**rectangle-closure [rect]**  
Return a closure that manages a single rectangle object. Calling the closure with no args returns the rectangle object. Otherwise, the messages #:w, #:h, #:x and #:y return the rectangle’s width, height, horizontal offset and vertical offset, respectively; and the messages #:w!, #:h!, #:x! and #:y!, followed by an integer, update the rectangle’s width, height, horizontal offset and vertical offset, respectively.

Optional arg rect specifies a rectangle object to manage instead of allocating a new one.

**rectangle<-geometry-string spec**  
Return a rectangle made from parsing the geometry string spec, which typically has the form \(WxH+X+Y\), where \(+X+Y\) is optional (defaults to “+0+0”), and \(W, H, X\) and \(Y\) are integers. Actually, the + can also be a -. If spec cannot be parsed, return #f. Examples:

```scheme
(rectangle<-geometry-string "42x43+44+45")
```
⇒ #<SDL-Rect 42x43+44+45>

(rectangle<-geometry-string "42x43-10-20")
⇒ #<SDL-Rect 42x43+-10+-20>

(rectangle<-geometry-string "42x43")
⇒ #<SDL-Rect 42x43+0+0>

(rectangle<-geometry-string "42")
⇒ #f

Note that the print representation of a rectangle always has “+”. The term “geometry string” derives from the X Window System, where many programs take a --geometry (or -g for short) command-line option.

poll-with-push-on-timeout-proc timeout slice [get-timeout-events] [Procedure]
Return a procedure P that checks the event queue for timeout ms, polling every slice ms. If an event arrives during that time, return #t. Otherwise return #f. Optional arg get-timeout-events is either a list of events to be pushed on the queue in the case of timeout, or a thunk to be called that produces such a list. If get-timeout-events is specified, return the result of another event queue polling. (This may still be #f if the pushed events are masked in some way.)

P is called with a single arg, a pre-constructed event object. This interface is congruent with that of wait-event and poll-event. See Chapter 4 [Events], page 13.

rect<-surface surface [x y] [Procedure]
Return a new rectangle with the same width and height as surface. Optional second and third arg (which must appear together or not at all) specifies the x and y components, respectively, to use instead of the default of 0 (zero).

(copy-rectangle rect)
⇒ #<SDL-Rect 3x4+1+2>

(copy-rectangle rect #:xy 11 22)
⇒ #<SDL-Rect 33x44+11+22>

(copy-rectangle rect #:wh 33 44)
⇒ #<SDL-Rect 33x44+1+2>

[Procedure]
Create a new surface and blit surface onto it. The new surface has the same pixel format as surface. Return the new surface.

Optional second arg clip is a rectangle describing the portion of surface to copy (default is the entire surface).
ignore-all-event-types-except [types...]  [Procedure]
Arrange to ignore all event types except those in types (see [event-type enums], page 45). As a special case, if types is #f, arrange to not ignore any event types (all are enabled).

In the following procs, those named ending with /3p return three values, each a thunk (unless specified otherwise) handling the three-phase calling convention, namely init, next, and done.

    (call-with-values (lambda () (foo/3p ...))
       (lambda (init! foo! done!)
         (init!)
         (let loop ((continue? (foo!)))
           (and continue? (loop (foo!))))
         (done!)))

Note that foo! returns non-#f to indicate that the looping is not yet complete.

fader/3p sec realized location image replacement  [Procedure]
Return three values, each a thunk, that can be used to loop for sec seconds, blitting onto realized at location (a rectangle or #f to indicate the origin) the alpha-composition of image and its replacement (both surfaces), to effect a fade-in of replacement over image. The alpha value is directly proportional to the time between the “next!” phase call and the “init!” phase call.

realized may be either a surface, in which case at the end of each loop it is shown via update-rect; or a pair whose CAR is a surface and whose CDR is a thunk that should do the showing.

Note that location is used for blitting, so its width and height should match those of image and replacement.

toroidal-panner/3p surface dx dy [sub [batch?]]  [Procedure]
Return three values, the first a procedure of one arg, the other two thunks, that can be used to toroidally pan surface by dx and dy pixels. This means that data disappearing from one side of the surface (left, right, top, bottom) is rotated to appear at the other side (right, left, bottom, top). The init! procedure takes one arg count, the number of pans to do.

Positive dx moves surface data to the left (panning right), and likewise, positive dy, up (panning down).

Optional third arg sub is a rectangle object specifying a subset of the surface. The default is to pan the entire surface.

Optional fourth arg batch? non-#f means to call update-rect on the (sub)surface after all the panning is done. The default is to update the surface after each pan. Batch mode is useful for implementing variable-speed panning, for example:

    (define (pan dir)
       (call-with-values (lambda () (toroidal-panner/3p screen
          (* dir 21)
          (* dir 12)
          #f #t))
        (lambda (init! next! done!)
           (lambda (count)
             (init! count)
             (let loop ((continue? (next!)))
               (and continue? (loop (next!))))
             (done!))))))
(define pan-away (pan 1))
(define pan-back (pan -1))
(define ramp (map 1+ (append (make-list 21 0)
                               (identity (iota 12))
                               (reverse! (iota 12))
                               (make-list 21 0))))

(for-each pan-away ramp)
(for-each pan-back ramp)
12 Simple Closures

This chapter documents module `(sdl simple)`. This module provides some simple abstractions to introduce common Guile-SDL programming idioms. Although the interfaces are documented, they are permanently alpha, that is, subject to change w/o notice. Instead of relying on the stability of the interface, you are encouraged to look at the implementation as a model for creating customized abstractions.

```scheme
simple-canvas init? w h bpp [flags...]
   [Procedure]
   Return a canvas closure that accepts a few simple messages. If init? is non-#f, initialize the SDL video subsystem first. w, h, and bpp specify the width, height, and bits-per-pixel, respectively. flags are symbols to set the video mode. If omitted, the default is hw-surface and doublebuf.

   The closure, if called without arguments, returns the video surface. Otherwise, the following messages are recognized:

   #:rect    Return a rectangle the width and height of the canvas.

   #:set-bg! r g b
   Set the background color (used for clearing) to the color specified by r, g and b (integers 0-255), respectively. By default it is black (all values zero).

   #:clear!  Fill the canvas with the background color.

   #:w
   #:h
   #:w/h   Return width, height, or a cons of width and height, respectively.

   #:resize! new-width new-height
   Request that the canvas dimension be changed to new-width by new-height. Return a rect that reflects the actual dimension.

simple-stylus init? filename size r g b
   [Procedure]
   Return a stylus closure that accepts a few simple messages. If init? is non-#f, initialize the SDL TTF support first. filename specifies the .ttf file to load and size the size. r, g and b are integers (0-255) specifying the color. The closure recognizes the following messages:

   #:set-font! filename size
   #:set-color! r g b
   Change the font or color, respectively.

   #:set-canvas! surface
   Set the surface on which the #:write! command renders.

   #:render text [color [bg]]
   Return a surface of text rendered using the default font, size, color and size. Optional second arg color specifies another color to use. Optional third arg bg specifies a background mode: #f (default) for “solid”; #t for “blended”; a color to use that color.

   #:write! where text [color [bg]]
   Similar to #:render, but also blit the surface onto the canvas at the rectangle position specified by where. The width and height components of where are updated by side effect.
```
simple-vpacked-image filename [canvas]

[Procedure]
Return a vpacked image closure that accepts a few simple messages. "Vpacked" means multiple vertically-abutted images of dimensions NxN (at the top) through Nx1 (at the bottom), stored in a single image file. filename specifies the file and optional arg canvas specifies a surface for blitting. The closure recognizes the following messages:

#:set-canvas! surface
  Change the canvas.

#:rects
  Return the vector of rectangles of length N+1 (the element at index zero is #f) corresponding to areas on the image representing the smaller sub-images. The element at index I is a rectangle of dimension IxI.

#:blit! i rect
  Blit the sub-image i (an integer 1 <= I <= N), onto the canvas. rect specifies a rectangle to blit to.
13 Excuses

Here are some notes on interface elements from `/usr/include/SDL/*.h` that are not yet wrapped by Guile-SDL. As things progress elements will be removed until an irreducible set remains.

Interface elements have zero or more attributes, some of which indicate irreducibility (such as `probably-never`). Following the attribute groupings are specific notes on those elements that are particular in some way. The presentation order is not significant.

13.1 Categories

For brevity, we omit the `SDL_` prefix in the groupings. There are two special cases: (N) stands for `SDLNet_`, and (M) stands for `Mix_`.

internal

These interface elements are exposed in the C header but should not be exposed to Scheme, for reasons of either safety or inutility.

```
SoftStretch  LowerBlit  UpperBlit
VideoInit   VideoQuit  AudioQuit  AudioInit
(M)GetChunk
```

probably-never

Don’t expect to see these exposed to Scheme, ever!

```
SoftStretch  SaveBMP_RW  LoadBMP_RW
VideoInit   VideoQuit   InitQuickDraw  RegisterApp
SetModuleHandle  getenv  putenv
ClearError  SetError  WriteBE64  WriteLE64
WriteBE32  WriteLE32  WriteBE16  WriteLE16
ReadBE64  ReadLE64  ReadBE32  ReadLE32
ReadBE16  ReadLE16  CloseAudio  UnlockAudio
LockAudio  MixAudio  ConvertAudio  BuildAudioCVT
FreeWAV  LoadWAV_RW  PauseAudio  GetAudioStatus
OpenAudio  AudioDriverName  AudioQuit
AudioInit  (M)GetMusicHookData  (M)GetChunk
```

doze

Windoze support, blech.

```
SaveBMP_RW  LoadBMP_RW  RegisterApp
SetModuleHandle
```

threading-implications

Will (any :-) ttn ever be ready for parallelism?

```
RemoveTimer  AddTimer  SetTimer  KillThread
WaitThread  GetThreadID  ThreadID
CreateThread  CondWaitTimeout  CondWait
CondBroadcast  CondSignal  DestroyCond
CreateCond  SemValue  SemPost  SemWaitTimeout
SemTryWait  SemWait  DestroySemaphore  CreateSemaphore
DestroyMutex  mutexV  mutexP  CreateMutex
```

todo

To be completed by Guile-SDL 1.0 (that is, if All Goes Well).

```
KillThread  WaitThread  GetThreadID
ThreadID  CreateThread  CondWaitTimeout
```
Chapter 13: Excuses

CondWait CondBroadcast CondSignal
DestroyCond CreateCond SemValue
SemPost SemWaitTimeout SemTryWait
SemWait DestroySemaphore CreateSemaphore
DestroyMutex mutexV mutexP CreateMutex
(N)Init (N)Quit (N)ResolveHost (N)ResolveIP
(N)TCP_Open (N)TCP_Accept (N)TCP_GetPeerAddress
(N)TCP_Send (N)TCP_Recv (N)TCP_Close
(N)AllocPacket (N)ResizePacket (N)FreePacket
(N)AllocPacketV (N)FreePacketV (N)UDP_Open
(N)UDP_Bind (N)UDP_Unbind (N)UDP_GetPeerAddress
(N)UDP_SendV (N)UDP_Send (N)UDP_RecvV
(N)UDP_Recv (N)UDP_Close (N)AllocSocketSet
(N)AddSocket (N)DelSocket (N)CheckSockets
(N)SocketReady (N)FreeSocketSet (N)Write16
(N)Write32 (N)Read16 (N)Read32 (M)SetPostMix
(M)HookMusic (M)HookMusicFinished (M)ChannelFinished
(M)RegisterEffect (M)UnregisterEffect (M)UnregisterAllEffects
(M)SetReverb (M)SetReverseStereo (M)SetMusicPosition
(M)SetSynchroValue (M)GetSynchroValue

rwops
Read-write operations.
FreeRW AllocRW RWFromMem RWFromConstMem
RWFromFile

macos
Macintosh support, meh.
InitQuickDraw

endian
These concern little- vs. big-endian i/o. Perhaps Guile already provides decent alternatives.
WriteBE64 WriteLE64 WriteBE32 WriteLE32
WriteBE16 WriteLE16 ReadBE64 ReadLE64
ReadBE32 ReadLE32 ReadBE16 ReadLE16

use-mixer-instead
These elements are obsoleted by the module (sdl mixer).
CloseAudio UnlockAudio LockAudio
MixAudio ConvertAudio BuildAudioCVT
FreeWAV LoadWAV_RW PauseAudio GetAudioStatus
OpenAudio AudioDriverName AudioQuit
AudioInit

hook
Callback from SDL to Scheme code. Can be tricky to get right...
(M)SetPostMix (M)HookMusic (M)HookMusicFinished (M)ChannelFinished
(M)RegisterEffect (M)UnregisterEffect (M)UnregisterAllEffects

13.2 Specific Notes
SDL_SoftStretch
SDL_video.h sez:
/* Not in public API at the moment - do not use! */

SDL_CreateRGBSurfaceFrom
not sure what this is useful for

SDL_GL_UpdateRects
arglist: (int numrects, SDL_Rect* rects)
we can either try to map uniform vectors (of smobs),
or introduce a 'RectVector' smob.

SDL_VideoInit
actually, SDL_video.h sez:
/* These functions are used internally, and should not be used unless you
* have a specific need to specify the video driver you want to use.
* You should normally use SDL_Init() or SDL_InitSubSystem().
* ...
*/

SDL_VideoQuit
see note for 'SDL_VideoInit'

SDL_Linked_Version
SDL_version.h sez:
/* This function gets the version of the dynamically linked SDL library.
it should NOT be used to fill a version structure, instead you should
use the SDL_Version() macro.
*/

SDL_GetWMInfo
return value for proc 'get-wm-info' does not presently
include the 'lock_func' and 'unlock_func' hooks.
support for those will be added after i figure out
how to "thunkify" them.

SDL_GetKeyName
why do we want to know the name of a key?

SDL_AudioQuit
SDL_audio.h sez:
/* These functions are used internally, and should not be used unless you
* have a specific need to specify the audio driver you want to use.
* You should normally use SDL_Init() or SDL_InitSubSystem().
*/

SDL_AudioInit
see note for 'SDL_AudioQuit'

SDLNet_AddSocket
there are also:
#define SDLNet_TCP_AddSocket
#define SDLNet_UDP_AddSocket

SDLNet_DelSocket
there are also:
#define SDLNet_TCP_DelSocket
#define SDLNet_UDP_DelSocket
Mix_GetMusicHookData
   If (when) ‘Mix_HookMusic’ is added, it will not support "user data".
   It’s better to use object properties for that.
Appendix A Stashes

There are 21 stashes (11 enums, 10 flags).

Distribution of symbols count:

<table>
<thead>
<tr>
<th>Count</th>
<th>Symbols</th>
<th>min:</th>
<th>max:</th>
<th>mean:</th>
<th>median:</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>5</td>
<td>2</td>
<td>231</td>
<td>17.0</td>
<td>5.0</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>231</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Distribution of symbol lengths:

<table>
<thead>
<tr>
<th>Length</th>
<th>Symbols</th>
<th>min:</th>
<th>max:</th>
<th>mean:</th>
<th>median:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>26</td>
<td>1</td>
<td>17</td>
<td>6.5126</td>
<td>7.0</td>
</tr>
<tr>
<td>2</td>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>23</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>38</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>28</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>28</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>103</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

activity-change
lost gained [2 enums]

alpha-limit
transparent opaque [2 enums]

application-state
mouseover inputfocus active [3 flags]

cd-track-type
audio data [2 enums]

cdrom-state
tray-empty playing error
stopped paused [5 enums]

event-mask
Note that these are a proper superset of those in the event-type enums, below.
### Appendix A: Stashes

<table>
<thead>
<tr>
<th>active</th>
<th>mouse-button-up</th>
<th>joy-button-up</th>
</tr>
</thead>
<tbody>
<tr>
<td>key-down</td>
<td>mouse</td>
<td>joy</td>
</tr>
<tr>
<td>key-up</td>
<td>joy-axis-motion</td>
<td>quit</td>
</tr>
<tr>
<td>key</td>
<td>joy-ball-motion</td>
<td>sys-wm</td>
</tr>
<tr>
<td>mouse-motion</td>
<td>joy-hat-motion</td>
<td>video-resize</td>
</tr>
<tr>
<td>mouse-motion</td>
<td>joy-button-down</td>
<td>video-expose</td>
</tr>
</tbody>
</table>

**event-type**

Note that these are a proper subset of those in the `event-mask` flags, above.

<table>
<thead>
<tr>
<th>active</th>
<th>mouse-button-up</th>
<th>joy-button-up</th>
</tr>
</thead>
<tbody>
<tr>
<td>key-down</td>
<td>mouse</td>
<td>joy</td>
</tr>
<tr>
<td>key-up</td>
<td>joy-axis-motion</td>
<td>quit</td>
</tr>
<tr>
<td>key</td>
<td>joy-ball-motion</td>
<td>sys-wm</td>
</tr>
<tr>
<td>mouse-motion</td>
<td>joy-hat-motion</td>
<td>video-resize</td>
</tr>
<tr>
<td>mouse-button-down</td>
<td>joy-button-down</td>
<td>video-expose</td>
</tr>
</tbody>
</table>

**fading-status**

| no | out | in |

**font-rotation**

<table>
<thead>
<tr>
<th>none</th>
<th>upside-down</th>
</tr>
</thead>
<tbody>
<tr>
<td>clockwise</td>
<td>counter-clockwise</td>
</tr>
</tbody>
</table>

**font-style**

<table>
<thead>
<tr>
<th>normal</th>
<th>italic</th>
<th>strikethrough</th>
</tr>
</thead>
<tbody>
<tr>
<td>bold</td>
<td>underline</td>
<td></td>
</tr>
</tbody>
</table>

**grab-mode**

| off | on     | query |

**init**

<table>
<thead>
<tr>
<th>timer</th>
<th>cdrom</th>
<th>no-parachute</th>
</tr>
</thead>
<tbody>
<tr>
<td>audio</td>
<td>joystick</td>
<td>event-thread</td>
</tr>
<tr>
<td>video</td>
<td>everything</td>
<td></td>
</tr>
</tbody>
</table>

**joystick-hat-position**

<table>
<thead>
<tr>
<th>centered</th>
<th>right</th>
<th>left</th>
</tr>
</thead>
<tbody>
<tr>
<td>up</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**keyboard-modifier**

<table>
<thead>
<tr>
<th>L-shift</th>
<th>L-alt</th>
<th>num</th>
</tr>
</thead>
<tbody>
<tr>
<td>R-shift</td>
<td>R-alt</td>
<td>caps</td>
</tr>
<tr>
<td>L-ctrl</td>
<td>L-meta</td>
<td>mode</td>
</tr>
<tr>
<td>R-ctrl</td>
<td>R-meta</td>
<td></td>
</tr>
</tbody>
</table>

**keyboard/button-state**

| released | pressed |

**keysym**

Note that digits begin with D- so that they are unambiguously (to read) symbols.

```
  a b c d e f g h i j k l m n o p q r s t u v w x y z
D-0 ... D-9
```
Appendix A: Stashes

<table>
<thead>
<tr>
<th>Character</th>
<th>Keys</th>
</tr>
</thead>
<tbody>
<tr>
<td>ampersand</td>
<td>backquote, capslock, delete, end</td>
</tr>
<tr>
<td>asterisk</td>
<td>backslash, caret, dollar, equals</td>
</tr>
<tr>
<td>at</td>
<td>backspace, clear, colon, comma, compose</td>
</tr>
<tr>
<td>break</td>
<td>break</td>
</tr>
<tr>
<td>f1 ... f15</td>
<td>f1 ... f15</td>
</tr>
<tr>
<td>hash</td>
<td>insert, help, home</td>
</tr>
<tr>
<td>kp-0 ... kp-9</td>
<td>kp-0 ... kp-9</td>
</tr>
<tr>
<td>kp-plus</td>
<td>kp-equals</td>
</tr>
<tr>
<td>kp-minus</td>
<td>kp-period</td>
</tr>
<tr>
<td>kp-multiply</td>
<td>kp-enter</td>
</tr>
<tr>
<td>kp-divide</td>
<td></td>
</tr>
<tr>
<td>left</td>
<td>right</td>
</tr>
<tr>
<td>L-alt L-ctrl L-meta L-shift L-super</td>
<td></td>
</tr>
<tr>
<td>R-alt R-ctrl R-meta R-shift R-super</td>
<td></td>
</tr>
<tr>
<td>L-bracket</td>
<td>R-bracket</td>
</tr>
<tr>
<td>L-paren</td>
<td>R-paren</td>
</tr>
<tr>
<td>less</td>
<td>greater</td>
</tr>
<tr>
<td>menu</td>
<td>pagedown, question, scrolllock, tab</td>
</tr>
<tr>
<td>minus</td>
<td>pageup, quote, semicolon</td>
</tr>
<tr>
<td>mode</td>
<td>pause, quotedbl, slash, underscore</td>
</tr>
<tr>
<td>numlock</td>
<td>period, space, undo</td>
</tr>
<tr>
<td>power</td>
<td>plus, return, sysreq</td>
</tr>
<tr>
<td>print</td>
<td></td>
</tr>
<tr>
<td>world-0 ... world-95</td>
<td>world-0 ... world-95</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mouse Button</th>
<th>[7 flags]</th>
</tr>
</thead>
<tbody>
<tr>
<td>left</td>
<td>wheel-up, x1</td>
</tr>
<tr>
<td>middle</td>
<td>wheel-down, x2</td>
</tr>
<tr>
<td>right</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mouse Button</th>
<th>[7 enums]</th>
</tr>
</thead>
<tbody>
<tr>
<td>left</td>
<td>wheel-up, x1</td>
</tr>
<tr>
<td>middle</td>
<td>wheel-down, x2</td>
</tr>
<tr>
<td>right</td>
<td></td>
</tr>
</tbody>
</table>
overlay  [5 flags]
Although these should be enums, these are putative flags due to a limitation in the implementation\(^1\). Pros that use them enforce enums-ish usage, anyway; a list of symbols results in an error.

<table>
<thead>
<tr>
<th>YV12</th>
<th>YYYU</th>
<th>UYVY</th>
</tr>
</thead>
<tbody>
<tr>
<td>YUY2</td>
<td>IYUV</td>
<td></td>
</tr>
</tbody>
</table>

palette  [2 flags]

| logical | physical |

video  [17 flags]

| sw-surface | no-frame | prealloc |
| hw-surface | hw-accel  | any-format |
| opengl     | src-colorkey | hw-palette |
| async-blit | rle-accel-ok | doublebuf |
| opengl-blit | rle-accel | fullscreen |
| resizable  | src-alpha |

\(^1\) For speed, we use immediate integers (aka fixnums) for enums, but those are not wide enough on a 32-bit system to hold the overlay values. Probably this should be rectified prior to release as it represents a semi-regression. OTOH, it’s not like anyone is actually using \texttt{create-yuv-overlay} anyway...
Appendix B  GNU Free Documentation License

Version 1.3, 3 November 2008
https://fsf.org/

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE
The purpose of this License is to make a manual, textbook, or other functional and useful document free in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.
This License is a kind of “copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.
We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS
This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The “Document”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as “you”. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.
A “Modified Version” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.
A “Secondary Section” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.
The “Invariant Sections” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.
The “Cover Texts” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.
A “Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input into text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “Opaque”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The “Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

The “publisher” means any person or entity that distributes copies of the Document to the public.

A section “Entitled XYZ” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “Acknowledgements”, “Dedications”, “Endorsements”, or “History”.) To “Preserve the Title” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document’s license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both
covers must also clearly and legibly identify you as the publisher of these copies. The front
cover must present the full title with all words of the title equally prominent and visible.
You may add other material on the covers in addition. Copying with changes limited to
the covers, as long as they preserve the title of the Document and satisfy these conditions,
can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the
first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto
adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you
must either include a machine-readable Transparent copy along with each Opaque copy,
or state in or with each Opaque copy a computer-network location from which the general
network-using public has access to download using public-standard network protocols a
complete Transparent copy of the Document, free of added material. If you use the latter
option, you must take reasonably prudent steps, when you begin distribution of Opaque
copies in quantity, to ensure that this Transparent copy will remain thus accessible at the
stated location until at least one year after the last time you distribute an Opaque copy
(directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before
redistributing any large number of copies, to give them a chance to provide you with an
updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions
of sections 2 and 3 above, provided that you release the Modified Version under precisely
this License, with the Modified Version filling the role of the Document, thus licensing
distribution and modification of the Modified Version to whoever possesses a copy of it. In
addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the
Document, and from those of previous versions (which should, if there were any, be
listed in the History section of the Document). You may use the same title as a previous
version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for
authorship of the modifications in the Modified Version, together with at least five of
the principal authors of the Document (all of its principal authors, if it has fewer than
five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the
publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copy-
right notices.

F. Include, immediately after the copyright notices, a license notice giving the public
permission to use the Modified Version under the terms of this License, in the form
shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover
Texts given in the Document’s license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled “History”, Preserve its Title, and add to it an item stating
at least the title, year, new authors, and publisher of the Modified Version as given
on the Title Page. If there is no section Entitled “History” in the Document, create
one stating the title, year, authors, and publisher of the Document as given on its
Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the “History” section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled “Acknowledgements” or “Dedications”, Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled “Endorsements”. Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled “Endorsements” or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section Entitled “Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.
In the combination, you must combine any sections Entitled “History” in the various original documents, forming one section Entitled “History”; likewise combine any sections Entitled “Acknowledgements”, and any sections Entitled “Dedications”. You must delete all sections Entitled “Endorsements.”

6. COLLECTIONS OF DOCUMENTS
You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS
A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION
Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled “Acknowledgements”, “Dedications”, or “History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION
You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.
Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See https://www.gnu.org/licenses/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License “or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy’s public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

“Massive Multiauthor Collaboration Site” (or “MMC Site”) means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A “Massive Multiauthor Collaboration” (or “MMC”) contained in the site means any set of copyrightable works thus published on the MMC site.

“CC-BY-SA” means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

“Incorporate” means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is “eligible for relicensing” if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.
ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright (C) year your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts. A copy of the license is included in the section entitled ‘‘GNU
Free Documentation License’’.

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the
"with...Texts." line with this:

with the Invariant Sections being list their titles, with
the Front-Cover Texts being list, and with the Back-Cover Texts
being list.

If you have Invariant Sections without Cover Texts, or some other combination of the three,
merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these
examples in parallel under your choice of free software license, such as the GNU General Public
License, to permit their use in free software.
Index

A
activity-change ........................................ 44
allocated-channels .................................... 25
alpha-limit ............................................. 44
application-state ..................................... 44

B
bit-rgba .................................................. 31
bit-surface ............................................. 11
button? ................................................... 18

c
caption-ti ............................................... 34
call-with-clip-rect .................................... 34
cd-close ................................................ 22
cd-eject .................................................. 22
cd-get-cur-frame ....................................... 21
cd-get-cur-frame ....................................... 21
cd-get-cur-track ....................................... 21
cd-get-nm-tracks ....................................... 21
cd-in-drive? ............................................. 21
cd-msf->frames ......................................... 22
cd-name ................................................... 21
cd-nth-track-itlo ...................................... 21
cd-num-drives ......................................... 21
cd-open ................................................... 21
cd-pause ................................................. 21
cd-play ................................................... 21
cd-play-tracks ......................................... 21
cd-resume .............................................. 21
cd-status ............................................... 21
cd-stop .................................................... 21
cd-track-type ......................................... 44
cd? ......................................................... 21
cdrom-state ........................................... 44
close-audio ............................................ 28
color:b ................................................... 7
color:g ................................................... 7
color:x ................................................... 7
color:set-b! ............................................. 8
color:set-g! ............................................. 8
color:set-r! ............................................. 7
color? ..................................................... 7
convert-surface ......................................... 11
copy-rectangle ......................................... 35
copy-surface ........................................... 35
create-cursor .......................................... 6
create-surface ......................................... 35
create-rgb-surface .................................. 10
create-rgb-square .................................... 34
create-rgb-surface .................................... 34
create-yuv-overlay .................................... 6

d
delay ........................................................ 5
device-fc .................................................. 25
display-format ......................................... 8
display-format-alpha .................................... 9
display-yuv-overlay .................................... 9
draw-aa-circle .......................................... 29
draw-aa-ellipse ........................................ 29
draw-aa-line ............................................ 29
draw-aa-polygon ........................................ 30
draw-aa-trigon .......................................... 30
draw-arc ............................................... 29
draw-bezier ............................................ 30
draw-character ......................................... 30
draw-circle ............................................. 29
draw-ellipse ........................................... 29
draw-hline .............................................. 29
draw-line ............................................... 29
draw-pie-slice .......................................... 29
draw-point .............................................. 29
draw-polygon ........................................... 30
draw-rectangle ......................................... 29
draw-rounded-rectangle ............................... 29
draw-string ............................................. 30
draw-textured-polygon ................................ 30
draw-thick-line ........................................ 29
draw-trigon ............................................. 30
draw-vline .............................................. 29

e
enable-key-repeat ....................................... 18
enable-unicode ......................................... 18
enum->number .......................................... 3
error key non-member-symbol ........................ 44
event:mask ............................................... 44
event-type .............................................. 45
event-type-handling .................................... 45
event:active:gain ...................................... 13
event:active:set-gain! ................................ 13
event:active:set-state! ................................ 13
event:active:state .................................... 13
event:button:button ................................... 14
event:button:button! .................................. 15
event:button:set-state! ................................ 15
event:button:set-x! ................................... 15
event:button:set-y! ................................... 15
event:button:state ..................................... 14
event:button:x ......................................... 15
event:button:y ......................................... 15
event:j-axis:axis ...................................... 15
event:j-axis:set-axis! ................................ 15
event:j-axis:set-value! ............................... 15
event:j-axis:set-which! ............................... 15
event:j-axis:value .................................... 15
event:j-axis:which .................................... 15
event:ball:ball ......................................... 16
event:ball:set-ball! ................................... 16
event:ball:set-which! ................................ 16
event:ball:set-xrel! .................................. 16
event:ball:set-yrel! .................................. 16
| Index | | |
| imfi-mmx? | 32 |
| imfi-mul-c | 33 |
| imfi-muldiv2 | 32 |
| imfi-muldiv4 | 32 |
| imfi-mulnor | 32 |
| imfi-mul | 32 |
| imfi-normalize-linear | 33 |
| imfi-not | 32 |
| imfi-sub | 32 |
| imfi-sub-c | 33 |
| init | 5, 45 |
| init-subsystem | 5 |
| joystick-ball-xy | 19 |
| joystick-close | 20 |
| joystick-get-axis | 19 |
| joystick-get-button | 20 |
| joystick-get-hat | 19 |
| joystick-hat-position | 45 |
| joystick-index | 19 |
| joystick-name | 19 |
| joystick-num-axes | 19 |
| joystick-num-balls | 19 |
| joystick-num-buttons | 19 |
| joystick-num-hats | 19 |
| joystick-open | 19 |
| joystick-opened? | 19 |
| joystick-polling | 19 |
| joystick-update | 19 |
| joystick? | 19 |
| keyboard-modifier | 45 |
| keyboard/button-state | 45 |
| keysym | 45 |
| kotk | 3 |
| list-modes | 6 |
| load-bmp | 11 |
| load-font | 24 |
| load-image | 11 |
| load-music | 25 |
| load-wave | 25 |
| lock-surface | 11 |
| lock-yuv-overlay | 9 |

**M**

| make-color | 7 |
| make-event | 13 |
| make-fps-manager | 31 |
| make-rect | 6 |
| make-surface | 10 |
| map-rgb | 8 |
| map-rgba | 8 |
| mouse-button | 46 |
| mouse-bzy | 18 |
| multiply-pixel-alpha! | 31 |
| music-volume | 26 |
| must-lock? | 12 |

**N**

| non-member-symbol, error key | 3 |
| num-joysticks | 19 |
| number>enum | 3 |
| number>flags | 3 |

**O**

| open-audio | 25 |
| overlay | 47 |

**P**

| palette | 47 |
| pause | 27 |
| pause-music | 27 |
| paused-music? | 27 |
| paused? | 27 |
| pixel-rgb | 8 |
| pixel-rgba | 8 |
| play-channel | 26 |
| play-music | 26 |
| playing-music? | 27 |
| playing? | 27 |
| poll-event | 17 |
| poll-with-push-on-timeout-proc | 35 |
| pump-events | 17 |
| push-event | 17 |

**Q**

| quit | 5 |
| quit-subsystem | 5 |
Index

R
rect:h ................................................. 7
rect:set-h! ...................................... 7
rect:set-w! ...................................... 7
rect:set-x! ...................................... 7
rect:set-y! ...................................... 7
rect:w ............................................. 7
rect:x ............................................. 6
rect:y ............................................. 7
rect:surface ................................... 35
rectangle-closure ................................ 34
rectangle-geometry-string .................... 34
render-glyph ..................................... 24
render-text ..................................... 24
render-utf8 ..................................... 24
reserve-channels ................................ 25
resume ............................................ 27
resume-music .................................... 27
rewind-music .................................... 27
rotate-square ................................... 34
roto-zoom-surface ................................ 30
roto-zoom-surface-xy .......................... 30

S
s16 .................................................. 3
save-bmp ......................................... 11
set-caption ...................................... 9
set-clip-rect! ................................... 11
set-colors! ...................................... 8
set-cursor ....................................... 9
set-distance .................................... 27
set-event-filter ................................ 17
set-gamma ....................................... 8
set-gamma-ramp ................................ 8
set-icon .......................................... 9
set-mod-state .................................. 18
set-music-command ................................ 27
set-palette ..................................... 8
set-panning ..................................... 27
set-pixel-alpha! ................................ 31
set-position .................................... 28
set-video-mode .................................. 6
show-cursor ...................................... 9
shrink-surface ................................... 31
simple-canvas .................................... 38
simple-stylus .................................... 38
simple-vpacked-image ............................ 39
surface-alpha! ................................... 11
surface-color-key! ................................ 11
surface-get-format ................................ 10
surface-pixels .................................... 12
surface:depth .................................... 10
surface:flags ..................................... 10
surface:h .......................................... 10
surface:w .......................................... 10
surface? ......................................... 10

T
text-wh .......................................... 24
toggle-full-screen ................................ 10
toroidal-panner/3p ............................... 36
ttf-init .......................................... 24
ttf-quit .......................................... 24

U
u16 .................................................... 3
u8 .................................................... 3
uniform vector argument(s) .................... 3
unlock-surface ................................... 11
unlock-yuv-overlay ................................ 9
update-rect ...................................... 7
update-rects ...................................... 7
utf8-wh ........................................... 24

V
vertical-flip-surface ............................. 11
vh-flip-surface .................................. 11
video .............................................. 47
video-cmf ......................................... 6
video-driver-name ................................ 6
video-mode-ok ................................... 26
volume ............................................ 26

W
wait-event ......................................... 17
warp-mouse ....................................... 9
was-init .......................................... 5

Z
zoom-surface ..................................... 30