SDL2 references
for eForth Windows

version 1.1 - dimanche 3 novembre 2024

Autor

- Marc PETREMANN

Contents

Autor....1

SDL2....3

CreateRenderer   window index flag -- render....3
CreateWindow   zstr x y w h fl -- win....3
Delay   ms -- fl....4
DestroyRenderer   render -- fl....4
DestroyWindow   win -- fl....4
GetError   -- n....4
GetTicks   -- ms....4
GetWindowFlags   window -- win-flag....4
GetWindowSize   windows *w *h -- fl....4
GetWindowSizeInPixels   windows *w *h -- fl....5
HideWindow   window -- fl....5
Init   n -- n....5
LoadBMP   file -- surface....6
LoadBMP_RW   RWops freesrc -- 0|surface....6
MaximizeWindow   window -- fl....6
MinimizeWindow   windows -- fl....6
Quit   --....6
RaiseWindow   window -- fl....7
RenderClear   render -- 0|err....7
RenderDrawLine   render x0 y0 x1 y1 -- 0|err....7
RenderDrawLines   render *points count -- 0|err....7
RenderDrawRect   render *rect --....8
RenderDrawRects   render *rect count....8
RenderFillRect   render *rect -- 0|err....9
RenderGetLogicalSize   render *width *height -- 0|err....9
RenderPresent   render --....10
RenderSetLogicalSize   render w h -- 0|err....10
RestoreWindow   window -- fl....10
RWFromFile   file mode -- 0|RWops....10
SDL2.dll   -- <name>....11
SDL_Color   -- n....11
SDL_Color!   r g b addr --....11
SDL_INIT_VIDEO   -- n....11
SetRenderDrawColor   renderer r g b a -- fl....11
SetWindowSize   window w h --....11
ShowWindow   window -- fl....12

SDL2

CreateRenderer   window index flag -- render

Create a 2D rendering context for a window.

Parameters:

- **window** the window where rendering is displayed
- **index** the index of the rendering driver to initialize, or -1 to initialize the first one supporting the requested flags
- flags 0, or one or more SDL_RendererFlags OR'd together.

```
variable WIN0

z" My first window with SDL2"
    X0_SCREEN_POSITION Y0_SCREEN_POSITION SCREEN_WIDTH SCREEN_HEIGHT
    SDL_WINDOW_SHOWN   SDL.CreateWindow  WIN0 !

variable REN0

WIN0 @ -1 0 CreateRenderer  REN0 !
```

CreateWindow   zstr x y w h fl -- win

Create a window with the specified position, dimensions, and flags.

Parameters:

- **title** the title of the window, in UTF-8 encoding
- **x** the x position of the window, SDL_WINDOWPOS_CENTERED, or SDL_WINDOWPOS_UNDEFINED
- **y** the y position of the window, SDL_WINDOWPOS_CENTERED, or SDL_WINDOWPOS_UNDEFINED
- **w** the width of the window, in screen coordinates
- **h** the height of the window, in screen coordinates
- **flags** 0, or one or more SDL_WindowFlags OR'd together

Returns **win** that was created or 0 on failure.

```
\ define size and position for SDL window
800 constant SCREEN_WIDTH
400 constant SCREEN_HEIGHT
```

```
200 constant X0_SCREEN_POSITION
 50 constant Y0_SCREEN_POSITION

z" My first window with SDL2"
    X0_SCREEN_POSITION Y0_SCREEN_POSITION
    SCREEN_WIDTH SCREEN_HEIGHT
    SDL_WINDOW_SHOWN CreateWindow
    value WIN0
```

Delay   ms -- fl

Wait a specified number of milliseconds.

DestroyRenderer   render -- fl

Destroy the rendering context for a window and free associated textures.

```
\ free ressources, end renderer and window
: freeRessources ( -- )
    REN0 DestroyRenderer drop
    WIN0 DestroyWindow   drop
    Quit
  ;
```

DestroyWindow   win -- fl

Destroy a window.

```
\ WIN0 must be declared by value and set by CreateWindow
WIN0 DestroyWindow
```

GetError   -- n

Retrieve a message about the last error that occurred on the current thread.

GetTicks   -- ms

Get the number of milliseconds since SDL library initialization.

GetWindowFlags   window -- win-flag

Get the window flags.

GetWindowSize   windows *w *h -- fl

Get the size of a window's client area.

Parameters:

- **window** the window to query the width and height from.

- **w** a pointer filled in with the width of the window, in screen coordinates
- **h** a pointer filled in with the height of the window, in screen coordinates

GetWindowSizeInPixels   windows *w *h -- fl

Get the size of a window in pixels.

Parameters:

- **window** the window from which the drawable size should be queried
- **w** a pointer to variable for storing the width in pixels
- **h** a pointer to variable for storing the height in pixels

```
variable WIN.width
variable WIN.height
: draw ( -- )
    REN0 255 255 255 255 SetRenderDrawColor drop
    REN0 RenderClear drop
    REN0 RenderPresent drop
    WIN0 WIN.width WIN.height GetWindowSizeInPixels drop
  ;
draw
WIN.width  @ .    \ display: 800
WIN.height @ .    \ display: 400
```

HideWindow   window -- fl

Hide a window.

Init   n -- n

Initialize the SDL library.

n must be one of

SDL_INIT_TIMER \ timer subsystem

SDL_INIT_AUDIO \ audio subsystem

SDL_INIT_VIDEO \ video subsystem; automatically initializes the events subsystem

SDL_INIT_JOYSTICK \ joystick subsystem; automatically initializes the events subsystem

SDL_INIT_HAPTIC \ haptic (force feedback) subsystem

SDL_INIT_GAMECONTROLLER \ controller subsystem; automatically initializes the joystick subsystem

SDL_INIT_EVENTS \ events subsystem

SDL_INIT_SENSOR

Returns 0 on success or a negative error code on failure. Call `GetError` for more information.

```
\ Initialize SDL with error management
: SDL.init ( n -- )
    Init
    if
        ." SDl could not intialize! SDL_Error: " getError .
    then
  ;
SDL_INIT_VIDEO SDL.init
```

LoadBMP   file -- surface

Load a surface from a file.

Do not use this word directly, but `SDL.load-image`.

```
: LoadBMP  ( zstr -- 0|surface )
    z" rb" RWFromFile 1 LoadBMP_RW
  ;

: SDL.load-image  ( zstr -- surface )
    LoadBMP ?dup 0= if
        drop -1 SDL.error
    then
  ;
```

LoadBMP_RW   RWops freesrc -- 0|surface

Load a BMP image from a seekable SDL data stream.

Parameters:

- **RWops** the data stream for the surface
- **freesrc** non-zero to close the stream after being read.

Returns a pointer to a new SDL_Surface structure or NULL if there was an error.

MaximizeWindow   window -- fl

Make a window as large as possible.

MinimizeWindow   windows -- fl

Minimize a window to an iconic representation.

Quit   --

Clean up all initialized subsystems.

RaiseWindow   window -- fl

Raise a window above other windows and set the input focus.

RenderClear   render -- 0|err

Clear the current rendering target with the drawing color.

```
\ REN0 is a value previsously initialized with CreateRenderer
REN0 RenderClear drop
```

RenderDrawLine   render x0 y0 x1 y1 -- 0|err

Draw a line on the current rendering target.

Parameters:

- **renderer** the rendering context
- **x1** the x coordinate of the start point
- **y1** the y coordinate of the start point
- **x2** the x coordinate of the end point
- **y2** the y coordinate of the end point

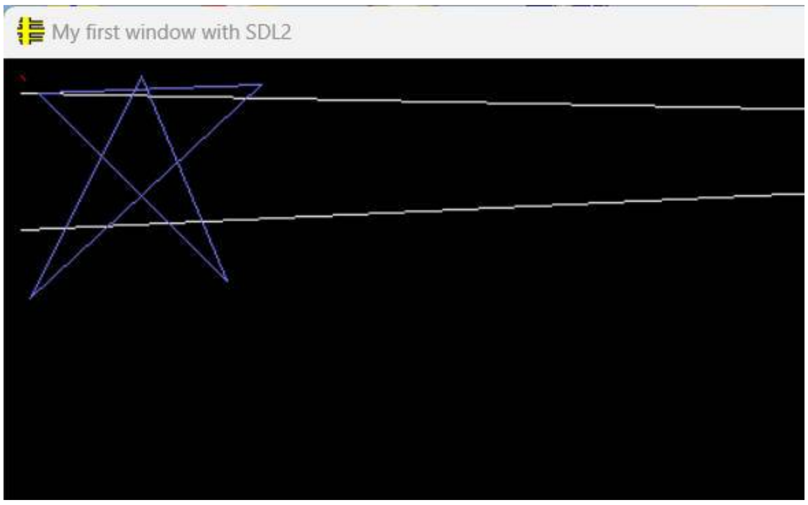
Returns 0 on success or a negative error code on failure.

```
\ color to white - draw simple line
REN0 255 255 255 255 SetRenderDrawColor drop
REN0 10 20 1200 45 RenderDrawLine drop
REN0 1200 45 10 100 RenderDrawLine drop
REN0 RenderPresent drop
```

RenderDrawLines   render *points count -- 0|err

Draw a series of connected lines on the current rendering target.

```
6 constant STAR_COUNT
create STAR_POINTS
     20 L,  20 L,    150 L,  15 L,
     15 L, 140 L,     80 L,  10 L,
    130 L, 130 L,     20 L,  20 L,
REN0 STAR_POINTS STAR_COUNT RenderDrawLines drop
REN0 RenderPresent drop
```



RenderDrawRect render *rect --

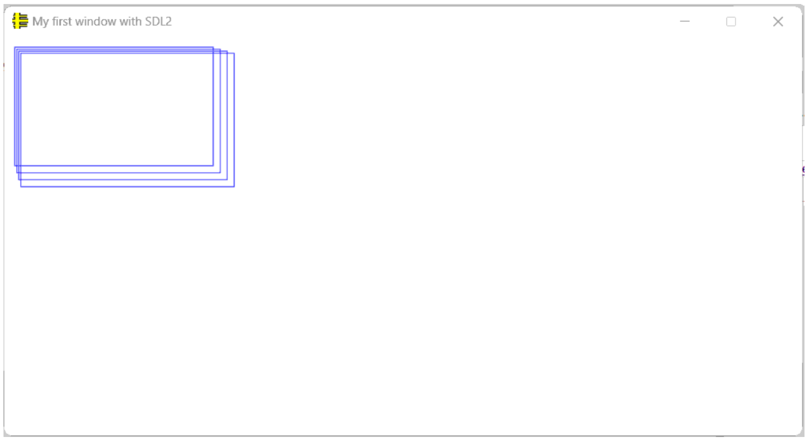
Draw a rectangle on the current rendering target.

```
create RECT
     10 L,   10 L,  200 L,  120 L,

REN0 63 63 255 255 SetRenderDrawColor drop
REN0 RECTS RenderDrawRect drop
REN0 RenderPresent drop
```

RenderDrawRects render *rect count

Draw some number of rectangles on the current rendering target.

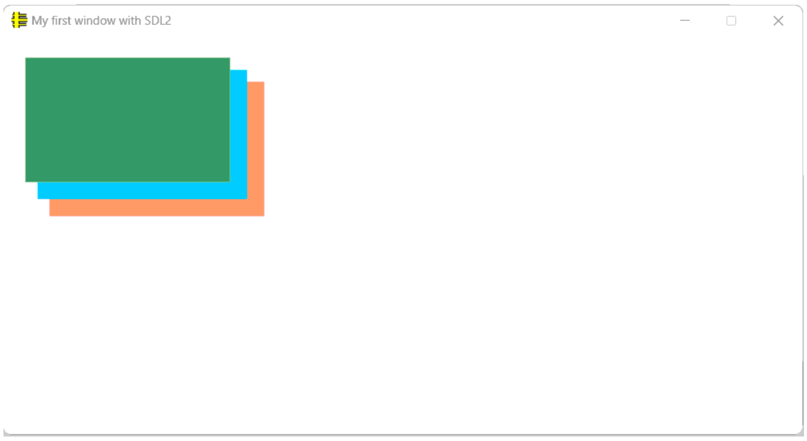
```
4 constant RECT_COUNT
create RECTS
     10 L,   10 L,  200 L,  120 L,
     12 L,   12 L,  205 L,  125 L,
     14 L,   14 L,  210 L,  130 L,
     16 L,   16 L,  215 L,  135 L,

REN0 63 63 255 255 SetRenderDrawColor drop
REN0 RECTS RECT_COUNT RenderDrawRects drop
REN0 RenderPresent drop
```



RenderFillRect   render *rect -- 0|err

Fill a rectangle on the current rendering target with the drawing color.

```
create RECTS
     10 L,   10 L,  200 L,  120 L,
     22 L,   22 L,  205 L,  125 L,
     34 L,   34 L,  210 L,  130 L,
     46 L,   46 L,  215 L,  135 L,

REN0 $FF $7D $5A $FF SetRenderDrawColor drop
REN0 3 RECTS ->SDL_Rect RenderFillRect drop

REN0 $45 $DC $FF $FF SetRenderDrawColor drop
REN0 2 RECTS ->SDL_Rect RenderFillRect drop

REN0 $45 $A3 $5E $FF SetRenderDrawColor drop
REN0 1 RECTS ->SDL_Rect RenderFillRect drop

REN0 RenderPresent drop
```



RenderGetLogicalSize   render *width *height -- 0|err

Get device independent resolution for rendering.

Only works if `RenderSetLogicalSize` was used. Otherwise stores 0.

```
variable REN0.width
variable REN0.height

REN0 REN0.width REN0.height RenderGetLogicalSize drop
```

RenderPresent   render --

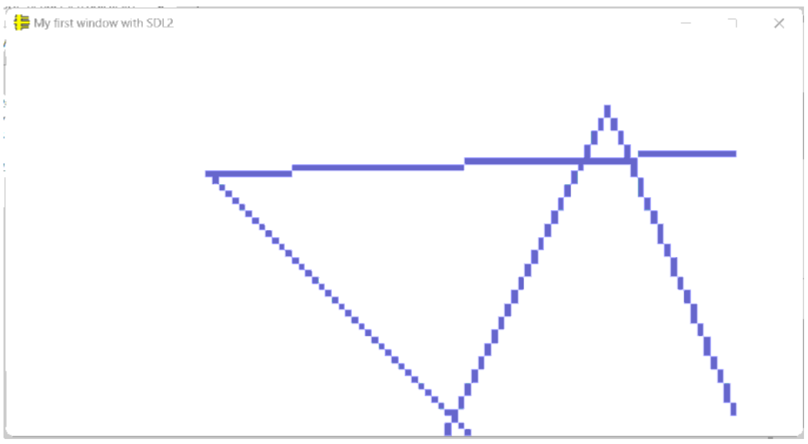
Update the screen with any rendering performed since the previous call.

RenderSetLogicalSize   render w h -- 0|err

Set a device independent resolution for rendering.

This function uses the viewport and scaling functionality to allow a fixed logical resolution for rendering, regardless of the actual output resolution. If the actual output resolution doesn't have the same aspect ratio the output rendering will be centered within the output display.

```
REN0 100 60 RenderSetLogicalSize drop
```



RestoreWindow   window -- fl

Restore the size and position of a minimized or maximized window.

RWFromFile   file mode -- 0|RWops

Use this word to create a new SDL_RWops structure for reading from and/or writing to a named file.

```
: LoadBMP   ( zstr -- 0|surface )
    z" rb" RWFromFile 1 LoadBMP_RW
  ;
```

SDL2.dll -- <name>

Entry point to SDL2.dll library

```
\ Destroy a window.
z" SDL_DestroyWindow"       1 SDL2.dll DestroyWindow ( window -- )
```

SDL_Color -- n

Sets the space required for an SDL_Color structure.

```
create border-color SDL_Color allot
  $ff $00 $00 border-color SDL_Color!
```

SDL_Color! r g b addr --

Assigns r g b color values in an `SDL_Color` structure.

```
create border-color SDL_Color allot
  $ff $00 $00 border-color SDL_Color!
```

SDL_INIT_VIDEO -- n

Constant. Tells the SDL that you want to initialize the video subsystem.

```
SDL_INIT_VIDEO SDL.Init
```

SetRenderDrawColor renderer r g b a -- fl

Set the color used for drawing operations (Rect, Line and Clear)

Parameters

- **renderer** the rendering context
- **r** the red value used to draw on the rendering target
- **g** the green value used to draw on the rendering target
- **b** the blue value used to draw on the rendering target
- **a** the alpha value used to draw on the rendering target; usually `SDL_ALPHA_OPAQUE` (255). Use `SetRenderDrawBlendMode` to specify how the alpha channel is used

SetWindowSize window w h --

Set the size of a window's client area.

Parameters:

- **window** the window to change
- **w** the width of the window in pixels, in screen coordinates, must be > 0/li>
- **h** the height of the window in pixels, in screen coordinates, must be > 0/li>

The window size in screen coordinates may differ from the size in pixels. Use `GetRendererOutputSize` to get the real client area size in pixels.

```
WIN0 400 200 SetWindowSize drop
```

ShowWindow window -- fl

Show a window.