Package: tigerfb (via r-universe)

July 6, 2026

Type Package

Title Tiny Graphics Engine for an R Frame Buffer

Version 1.0.0

Maintainer Mike Cheng <mikefc@coolbutuseless.com>

Description A cross-platform graphics device which offers simple update via pre-rendered images with keyboard and mouse feedback.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

SystemRequirements GNU make

LinkingTo colorfast

Depends R (>= 4.1.0)

Imports colorfast

Suggests nara

Config/roxygen2/version 8.0.0

Copyright This package is based in a large part on the Public Domain code packaged as the 'tigr' C library by Erik Agsjo. See COPYRIGHTS file for the public domain notice for this package.

Config/pak/sysreqs make

Repository https://coolbutuseless.r-universe.dev

Date/Publication 2026-07-06 21:39:48 UTC

RemoteUrl https://github.com/coolbutuseless/tigerfb

RemoteRef HEAD

RemoteSha cc91b9ba3a40118e5aa3a4d7d910a7c200959312

Contents

`fb_capture` *Capture the current window contents to a native raster image*

Description

Capture the current window contents to a native raster image

Usage

```
fb_capture(window)
```

Arguments

`window`	window handle as created by `fb_open()`

Value

A native raster image

See Also

Other capturing functions: `fb_save_png()`

Examples

```
window <- fb_open()
fb_rect(window, 10, 10, 70, 30, 'hotpink', 'blue')
fb_update(window)
nr <- fb_capture(window)
Sys.sleep(3)
fb_close(window)
class(nr)
```

`fb_close` *Destroy window in a safe manner.*

Description

This will reconnect to the specified window and prompt the user to close the window.

Usage

```
fb_close(window)
```

Arguments

`window`	window handle as created by `fb_open()`

Value

None

See Also

Other core window functions: `fb_open()`, `fb_update()`

Examples

```
window <- fb_open()
fb_rect(window, 10, 10, 70, 30, 'hotpink', 'blue')
fb_update(window)
Sys.sleep(3)
fb_close(window)
```

fb_demo *Bubble universe demonstration.*

Description

This demo generates an animation of an evolving bubble universe.

Usage

```
fb_demo(...)
```

Arguments

...	arguments passed to fb_open()

Details

Press 'ESC' key to quit, and then close the window.

Value

None

Examples

```
fb_demo(mode = 'fullscreen')
```

fb_fill *Clear window*

Description

Clear window

Usage

```
fb_fill(window, fill)
```

Arguments

window	window handle as created by fb_open()
fill	Fill color. Default: NA (transparent)

Value

None

See Also

Other drawing commands: fb_line(), fb_rect(), fb_text()

Examples

```
window <- fb_open(100, 100)
fb_fill(window, 'hotpink')
fb_update(window)
Sys.sleep(3)
fb_close(window)
```

---

fb_is_open *Check if a window is currently open and active*

Description

Check if a window is currently open and active

Usage

```
fb_is_open(window)
```

Arguments

window	window handle as created by fb_open()

Value

Logical value

Examples

```
window <- fb_open()
fb_is_open(window)
fb_close(window)
fb_is_open(window)
```

fb_key_names *Get the list of key names which are returned by* fb_state()

Description

Get the list of key names which are returned by fb_state()

Usage

```
fb_key_names()
```

Value

character vector

fb_line *Draw Line*

Description

Draw Line

Usage

```
fb_line(window, x1, y1, x2, y2, color = "black")
```

Arguments

window	window handle as created by fb_open()
x1, y1, x2, y2	line coordinates
color	Color. Default: 'black'

Value

None

See Also

Other drawing commands: fb_fill(), fb_rect(), fb_text()

Examples

```
window <- fb_open(100, 100)
fb_line(window, 0, 0, 50, 50, 'hotpink')
fb_update(window)
Sys.sleep(3)
fb_close(window)
```

fb_open *Initialise an interactive window*

Description

Initialise an interactive window

Usage

```
fb_open(
  width = 150,
  height = 100,
  mode = "auto",
  expand = 2,
  cursor = TRUE,
  title = "R"
)
```

Arguments

width, height	Initial window dimensions (in pixels)
mode	Window mode. Valid values:'auto', 'fixed', 'fullscreen'. Default: "auto" resizes the underlying image canvas when image is resized. "fixed" always keeps the same sized image canvas, but applies an integer scaling factor to maximize the display within the window.
expand	Integer value to expand pixel size. Only valid when mode = "auto". Valid values: 1, 2, 3, 4. Default: 2
cursor	show the cursor? Default: TRUE
title	window title. Default: "R"

Value

Window handle

See Also

Other core window functions: fb_close(), fb_update()

Examples

```
window <- fb_open()
fb_rect(window, 10, 10, 70, 30, 'hotpink', 'blue')
fb_update(window)
Sys.sleep(3)
fb_close(window)
```

fb_rect *Draw rectangle*

Description

Draw rectangle

Usage

```
fb_rect(window, x, y, w, h, color = "black", fill = NA)
```

Arguments

window	window handle as created by fb_open()
x, y, w, h	location and dimensions of rectangle
color	Outline color. Default: 'black'
fill	Fill color. Default: NA (transparent)

Value

None

See Also

Other drawing commands: fb_fill(), fb_line(), fb_text()

Examples

```
window <- fb_open(100, 100)
fb_rect(window, 10, 10, 70, 30, 'hotpink', 'blue')
fb_update(window)
Sys.sleep(3)
fb_close(window)
```

fb_save_png *Capture the current window contents to a PNG file*

Description

Capture the current window contents to a PNG file

Usage

```
fb_save_png(window, filename)
```

Arguments

window	window handle as created by fb_open()
filename	PNG filename

Value

None

See Also

Other capturing functions: fb_capture()

Examples

```
window <- fb_open()
fb_rect(window, 10, 10, 70, 30, 'hotpink', 'blue')
fb_update(window)
png_file <- tempfile(fileext = ".png")
fb_save_png(window, png_file)
Sys.sleep(3)
fb_close(window)
```

---

fb_state *Get the state of the current window i.e. keys and mouse*

Description

Get the state of the current window i.e. keys and mouse

Usage

```
fb_state(window)
```

Arguments

window	window handle as created by fb_open()

Value

environment of 'mouse' and 'key' information

`mouse$coords` integer vector of (x,y) coordinates

`mouse$buttons` integer vector of the state of three mouse buttons. Value represents the state of the button. See ?press_state for details

`key` named integer vector of all keycodes. Value represents the state of the key. See ?press_state for details

Examples

```
window <- fb_open()
fb_rect(window, 10, 10, 70, 30, 'hotpink', 'blue')
fb_update(window)
for (i in seq(30)) {
  fb_update(window)
  state <- fb_state(window)
  print(state$mouse$coords)
  Sys.sleep(0.1)
}
print(state)
fb_close(window)
```

---

fb_text *Draw Text using built-in font*

Description

Draw Text using built-in font

Usage

```
fb_text(window, x, y, text, color = "white")
```

Arguments

window	window handle as created by fb_open()
x, y	position
text	string
color	Color. Default: 'white'

Value

None

See Also

Other drawing commands: fb_fill(), fb_line(), fb_rect()

Examples

```
window <- fb_open(100, 100)
fb_text(window, 6, 42, "Hello #RStats!")
fb_update(window)
Sys.sleep(3)
fb_close(window)
```

fb_update *Update window display*

Description

Execute any queued drawing commands. User can also provide a new native raster image which will replace the current window contents

Usage

```
fb_update(window, nr = NULL)
```

Arguments

window	window handle as created by fb_open()
nr	(optional) native raster image the same dimensions as the window. This will be rendered over the top of the current contents of the window. Default: NULL means to only process the drawing queue, and update the keyboard and mouse state.

Value

None

See Also

Other core window functions: fb_close(), fb_open()

Examples

```
window <- fb_open()
fb_rect(window, 10, 10, 70, 30, 'hotpink', 'blue')
fb_update(window)
Sys.sleep(3)
fb_close(window)
```

press_state *Status codes for mouse and keyboard*

Description

Keys and mouse buttons may be 4 different states - not pressed, held down, has just transitioned from not-pressed-to-pressed, and has just transitioned from pressed-to-not-pressed.

Usage

```
press_state
```

Details

There 4 possible key states returned by `fb_state()`. For simple keyboard/mouse scanning the user need only check if the state is non-zero to indicate that the key or mouse button has been pressed.

For some complex handling, the user-interface may need to watch for a transition event i.e. from-pressed-to-not-pressed, or from-not-pressed-to-pressed (e.g. keyboard control in Doom)

`0` = `INACTIVE` The key/button is not pressed

`1` = `DOWN` The key/button has just now been pressed down

`2` = `HELD` The key/button is being held down

`3` = `UP` The key/button was being held but has just now been released

Examples

```
press_state[["INACTIVE"]]
press_state[["DOWN"]]
press_state[["HELD"]]
press_state[["UP"]]
```

---

print.tigerfb_state *More palatable printing of a 'state' environment*

Description

More palatable printing of a 'state' environment

Usage

```
## S3 method for class 'tigerfb_state'
print(x, ...)
```

Arguments

`x`	state object returned by `fb_state()`
`...`	ignored

Value

original x

Index