Package: skiagd (via r-universe)

March 24, 2025

Title R wrapper for 'rust-skia'

Version 0.0.0.9000

Description A toy R wrapper for 'rust-skia' <https://github.com/rust-skia/rust-skia> (the Rust crate 'skia_safe' <https://rust-skia.github.io/doc/skia_safe/>, a binding for 'Skia' <https://skia.org/>).

License MIT + file LICENSE

Depends R (>= 4.2)

Imports grDevices, grid, purrr, rlang

Suggests magick, testthat (>= 3.0.0), vdiffr

Config/testthat/edition 3

Encoding UTF-8

OS_type unix

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

SystemRequirements GNU make, Cargo (Rust's package manager), rustc, fontconfig, freetype2

Config/pak/sysreqs libfontconfig1-dev libfreetype6-dev make

Repository https://paithiov909.r-universe.dev

RemoteUrl https://github.com/paithiov909/skiagd

RemoteRef HEAD

RemoteSha 803e758e65ca5888a8b74ace5d7ec3ac071aa26a

Contents

`add_circle` *Add circles*

Description

Add circles

Usage

```
add_circle(img, center, radius, props = paint())
```

Arguments

`img`	A raw vector of picture.
`center`	A double matrix where each row is circle center.
`radius`	Numerics of circle radius.
`props`	A paint properties out of `paint()`.

Value

A raw vector of picture.

add_diff_rect *Add difference rectangles*

Description

Add difference rectangles

Usage

```
add_diff_rect(img, outer, outer_radii, inner, inner_radii, props = paint())
```

Arguments

img	A raw vector of picture.
outer	A double matrix where each row is an outer rectangle XYWH (left, top, right, bottom).
outer_radii	A double matrix where each row is a pair of axis lengths on X-axis and Y-axis of outer oval describing rounded corners.
inner	A double matrix where each row is an inner rectangle XYWH (left, top, right, bottom).
inner_radii	A double matrix where each row is a pair of axis lengths on X-axis and Y-axis of inner oval describing rounded corners.
props	A paint properties out of paint().

Value

A raw vector of picture.

add_line *Add lines*

Description

Add lines

Usage

```
add_line(img, from, to, props = paint())
```

Arguments

img	A raw vector of picture.
from	A double matrix where each row is a start point.
to	A double matrix where each row is an end point.
props	A paint properties out of paint().

Value

A raw vector of picture.

`add_path` *Add paths*

Description

Add paths

Usage

```
add_path(img, path, transform = c(1, 0, 0, 0, 1, 0, 0, 0, 1), props = paint())
```

Arguments

`img`	A raw vector of picture.
`path`	Characters of SVG notations like `"M10 10 H 90 V 90 H 10 L 10 10"`.
`transform`	Numerics of length 9 to apply affine transformations to the path themselves. See transform-matrix for affine transformations.
`props`	A paint properties out of `paint()`.

Value

A raw vector of picture.

`add_png` *Add PNG image to canvas*

Description

Add PNG image to canvas

Usage

```
add_png(img, png, left = 0, top = 0, props = paint())
```

Arguments

`img`	A raw vector of picture.
`png`	A raw vector of PNG data.
`left`	Left offset for drawing PNG image.
`top`	Top offset for drawing PNG image.
`props`	A paint properties out of `paint()`.

Value

A raw vector of picture.

`add_point` *Add points*

Description

Add points

Usage

```
add_point(img, point, props = paint())
```

Arguments

`img`	A raw vector of picture.
`point`	A double matrix where each row is a point.
`props`	A paint properties out of `paint()`.

Value

A raw vector of picture.

`add_rect` *Add rectangles*

Description

Add rectangles

Usage

```
add_rect(img, rect, props = paint())
```

Arguments

`img`	A raw vector of picture.
`rect`	A double matrix where each row is a rectangle XYWH (left, top, right, bottom).
`props`	A paint properties out of `paint()`.

Value

A raw vector of picture.

add_rounded_rect *Add rounded rectangles*

Description

Add rounded rectangles

Usage

```
add_rounded_rect(img, rect, radii, props = paint())
```

Arguments

img	A raw vector of picture.
rect	A double matrix where each row is a rectangle XYWH (left, top, right, bottom).
radii	A double matrix where each row is a pair of axis lengths on X-axis and Y-axis of oval describing rounded corners.
props	A paint properties out of paint().

Value

A raw vector of picture.

as_png *Convert picture into PNG data*

Description

Convert picture into PNG data

Usage

```
as_png(img, props = paint())
```

Arguments

img	A raw vector of picture.
props	A paint properties out of paint().

Value

A raw vector of PNG data.

as_recordedplot *Convert a picture into a recorded plot*

Description

This is mainly for testing purposes.

Usage

```
as_recordedplot(img, props = paint())
```

Arguments

img	A raw vector of picture.
props	A paint properties out of paint().

Value

A recordedplot object. See grDevices::recordPlot() for details.

BlendMode *BlendMode (0-28)*

Description

BlendMode determines how source and destination colors are combined.

Usage

```
BlendMode
```

Format

An object of class BlendMode__bundle (inherits from savvy_skiagd__sealed) of length 29.

Details

The following blend modes are available in Skia:

1. Clear
2. Src
3. Dst
4. SrcOver
5. DstOver
6. SrcIn

7. DstIn
8. SrcOut
9. DstOut
10. SrcATop
11. DstATop
12. Xor
13. Plus
14. Modulate
15. Screen
16. Overlay
17. Darken
18. Lighten
19. ColorDodge
20. ColorBurn
21. HardLight
22. SoftLight
23. Difference
24. Exclusion
25. Multiply
26. Hue
27. Saturation
28. Color
29. Luminosity

See Also

BlendMode in skia_safe - Rust

Other paint-attributes: Cap, FillType, Join, PathEffect, PointMode, Shader, Style

---

canvas *Create a canvas*

Description

Creates a new canvas filled with specified color.

Usage

```
canvas(fill = "transparent", size = paint()[["canvas_size"]])
```

Arguments

`fill`	RGBA representation of a color. This can be specified using named colors or hexadecimal color codes, which are converted internally using `grDevices::col2rgb()`.
`size`	Integers of length 2; canvas size.

Value

A raw vector of picture.

`Cap` *Cap (0-2)*

Description

`Cap` determines the stroke cap (the geometry drawn at the beginning and end of strokes).

Usage

```
Cap
```

Format

An object of class `Cap__bundle` (inherits from `savvy_skiagd__sealed`) of length 3.

Details

The following caps are available:

- `Butt`: Butt cap.
- `Round`: Round cap.
- `Square`: Square cap.

See Also

Cap in skia_safe::paint - Rust

Other paint-attributes: `BlendMode`, `FillType`, `Join`, `PathEffect`, `PointMode`, `Shader`, `Style`

col2rgba *Color to RGBA*

Description

A wrapper of grDevices::col2rgb().

Usage

```
col2rgba(color)
```

Arguments

color	col for grDevices::col2rgb().

Value

An integer vector.

dev_size *Device size*

Description

Just returns the size of the current device as an integer (not a numeric).

Usage

```
dev_size(units = "px")
```

Arguments

units	units for grDevices::dev.size().

Value

An integer vector.

draw_img *Plot picture as PNG image*

Description

Plot picture as PNG image

Usage

```
draw_img(img, props = paint())
```

Arguments

img	A raw vector of picture.
props	A paint properties out of paint().

Value

img is returned invisibly.

FillType *FillType (0-3)*

Description

FillType determines how paths are drawn. This is for add_path() only. Not used in other functions.

Usage

```
FillType
```

Format

An object of class FillType__bundle (inherits from savvy_skiagd__sealed) of length 4.

Details

The following FillType are available:

- Winding
- EvenOdd
- InverseWinding
- InverseEvenOdd

See Also

FillType in skia_safe::path - Rust

Other paint-attributes: `BlendMode`, `Cap`, `Join`, `PathEffect`, `PointMode`, `Shader`, `Style`

`freeze` *Freeze a picture*

Description

`as_png(img)` and then adds it to a new canvas.

Usage

```
freeze(img, fill = "transparent", props = paint())
```

Arguments

`img`	A raw vector of picture.
`fill`	A string scalar; named colors or hexadecimal color codes.
`props`	A paint properties out of `paint()`.

Value

A raw vector of picture.

`Join` *Join (0-2)*

Description

`Join` determines the stroke join (the geometry drawn at the corners of strokes) for shapes.

Usage

```
Join
```

Format

An object of class `Join__bundle` (inherits from `savvy_skiagd__sealed`) of length 3.

Details

The following joins are available:

- `Miter`: Miter join.
- `Round`: Round join.
- `Bevel`: Bevel join.

See Also

Join in skia_safe::paint - Rust

Other paint-attributes: BlendMode, Cap, FillType, PathEffect, PointMode, Shader, Style

---

paint *Define painting attributes*

Description

The paint() function allows users to specify various painting attributes for drawing shapes on the canvas, such as color, stroke width, and transformations.

Usage

```
paint(...)
```

Arguments

...	<dynamic-dots> Named arguments specifying painting attributes. See details.

Details

The following painting attributes can be specified:

- canvas_size: Integers of length 2 (width, height).
- color: RGBA representation of a color. This can be specified using named colors or hexadecimal color codes, which are converted internally using grDevices::col2rgb().
- style: The paint style. See Style.
- join: Stroke join. See Join.
- cap: Stroke cap. See Cap.
- width: A numeric scalar (stroke width).
- miter: A numeric scalar (stroke miter).
- blend_mode: See BlendMode.
- path_effect: See PathEffect.
- shader: See Shader.
- point_mode: PointMode for add_point().
- fill_type: FillType for add_path().
- transform: Numerics of length 9. See transform-matrix for affine transformations.

Value

A list containing the specified painting attributes, merged with default values.

PathEffect *PathEffect*

Description

`PathEffect` is a struct that offers a reference to `skia_safe::PathEffect`. You can apply a path effect to shapes via `paint()`. Currently only single `PathEffect` can be specified; multiple effects are not supported.

Arguments

`start`	A numeric scalar in the range `[0, 1]`.
`end`	A numeric scalar in the range `[0, 1]`.
`length`	A numeric scalar; length of the subsegments.
`deviation`	A numeric scalar; limit of the movement of the endpoints.
`seed`	An integer scalar; random seed.
`intervals`	A numeric vector; even number of entries with even indices specifying the length of the "on" intervals, and the odd index specifying the length of "off".
`phase`	A numeric scalar; offset into the intervals array (for `dash()`), or distance (mod advance) along the path for its initial position (for `path_1d()`).
`radius`	A numeric scalar; radius of the rounded corners.
`path`	A string scalar of SVG notation to replicate.
`advance`	A numeric scalar; space between instances of path.
`style`	A string scalar; how to transform path at each point. Can be `"translate"`, `"rotate"`, or `"morph"`.
`transform`	Numerics of length 9; transformation matrix.
`width`	A numeric scalar; width of the path to be stamped.

Details

The following effects are available:

- `no_effect()`: does not apply any path effect. This is the default effect for `paint()`.
- `trim(start, end)`: trims the `start` and `end` of the path. Note that you can't trim nothing at all, i.e., setting `start = 0` and `end = 1` arises an error.
- `discrete(lentgh, deviation, seed)`: applies discrete path effect.
- `dash(intervals, phase)`: applies dash path effect.
- `corner(radius)`: applies corner path effect.
- `path_1d(path, advance, phase, style)`: applies 1D path effect.
- `path_2d(path, transform)`: applies 2D path effect.
- `line_2d(width, transform)`: applies 2D line path effect.

Value

A PathEffect object.

See Also

Path Effects | React Native Skia

Other paint-attributes: BlendMode, Cap, FillType, Join, PointMode, Shader, Style

---

PointMode *PointMode (0-2)*

Description

PointMode determines how points are drawn. This is for add_point() only. Not used in other functions.

Usage

```
PointMode
```

Format

An object of class PointMode__bundle (inherits from savvy_skiagd__sealed) of length 3.

Details

The following PointMode are available:

- Points: Draws each point as a point. The shape of point drawn depends on props.
- Lines: Each pair of point draws a line segment. One line is drawn for every two points; each point is used once. If count is odd, the final point is ignored.
- Polygon: Each adjacent pair of point draws a line segment. count minus one lines are drawn; the first and last point are used once.

See Also

PointMode in skia_safe::canvas - Rust

Other paint-attributes: BlendMode, Cap, FillType, Join, PathEffect, Shader, Style

Shader *Shader*

Description

`Shader` is a struct that offers a reference to `skia_safe::Shader`. You can apply a shader to shapes via `paint()`.

Note that concatenating shaders with `c()` is equivalent to blend them all into a single shader using `Shader$blend()` with the default `BlendMode`. You can pass `mode` explicitly for `c()` to change the blend mode.

Arguments

`img`	A raw vector of picture.
`mode`	For `blend()`, BlendMode. For others, TileMode.
`tile_size`	Numerics of length 2; tile size (width, height).
`transform`	Numerics of length 9; transformation matrix.
`png_bytes`	A raw vector of PNG data.
`rgba`	Integers of length 4 in range `[0, 255]` representing RGBA.
`dst`	A `Shader` object; destination shader.
`src`	A `Shader` object; source shader.
`freq`	Numerics of length 2; frequencies.
`octaves`	A numeric scalar; number of octaves.
`seed`	Integer scalar; random seed.
`start`	Numerics of length 2; starting point (x, y).
`end`	Numerics of length 2; ending point (x, y).
`from`	Integers of lenth 4 in range `[0, 255]`; starting color.
`to`	Integers of length 4 in range `[0, 255]`; ending color.
`flags`	A logical scalar; typically, you can leave this as `FALSE`. See here for details.
`radii`	Numerics of length 2; radii of start and end circles.
`center`	Numerics of length 2; center of the gradient.
`start_angle`	A numeric scalar in range `[0, 360]`; starting angle. For default, set `0`.
`end_angle`	A numeric scalar in range `[0, 360]`; ending angle. For default, set `360`.

Details

The following shaders are available:

- `no_shader()`: does not apply any shader. This is the default shader for `paint()`.
- `from_picture(img, mode, tile_size, transform)`: takes a picture and returns an image shader.

- `from_png(png_bytes, mode, transform)`: takes a PNG image and returns an image shader.
- `color(rgba)`: takess a color and returns a color shader.
- `blend(mode, dst, src)`: returns a shader that combines the given shaders with a BlendMode.
- `fractal_noise(freq, octaves, seed, tile_size)`: fractal perlin noise shader.
- `turbulence(freq, octaves, seed, tile_size)`: turbulence noise shader.
- `linear_gradient(start, end, from, to, mode, flags, transform)`: linear gradient shader.
- `radial_gradient(center, radius, from, to, mode, flags, transform)`: radial gradient shader.
- `conical_gradient(start, end, radii, from, to, mode, flags, transform)`: conical gradient shader.
- `sweep_gradient(center, start_angle, end_angle, from, to, mode, flags, transform)`: sweep gradient shader.

Value

A `Shader` object.

See Also

- Image Shaders | React Native Skia
- Gradients | React Native Skia
- Perlin Noise Shaders | React Native Skia
- Blending and Colors | React Native Skia

Other paint-attributes: `BlendMode`, `Cap`, `FillType`, `Join`, `PathEffect`, `PointMode`, `Style`

`Style` *Style (0-2)*

Description

`Style` determines the stroke style of shapes.

Usage

```
Style
```

Format

An object of class `Style__bundle` (inherits from `savvy_skiagd__sealed`) of length 3.

Details

The following styles are available:

- `StrokeAndFill`: Stroke and fill.
- `Stroke`: Stroke only.
- `Fill`: Fill only.

See Also

Style in skia_safe::paint - Rust

Other paint-attributes: `BlendMode`, `Cap`, `FillType`, `Join`, `PathEffect`, `PointMode`, `Shader`

`TileMode` *TileMode (0-3)*

Description

`TileMode` determines how the source is tiled for shaders. This is not a paint attribute. To specify `TileMode`, directly pass these pointers to shader functions.

Usage

```
TileMode
```

Format

An object of class `TileMode__bundle` (inherits from `savvy_skiagd__sealed`) of length 4.

Details

The following `TileMode` are available:

- `Clamp`
- `Repeat`
- `Mirror`
- `Decal`

See Also

TileMode in skia_safe - Rust

`transform-matrix` *Applying affine transformations to a picture*

Description

When loading a picture into a canvas, you can apply an affine transformation by providing a numeric vector of length 9 to `paint()` as `transform`.

Details

This vector defines a transformation matrix that modifies a picture before rendering it onto the canvas.

The `transform` vector represents a 3x3 matrix used for affine transformations, following the format:

$$\begin{bmatrix} \text{scale}_x & \text{skew}_y & \text{pers}_0 \\ \text{skew}_x & \text{scale}_y & \text{pers}_1 \\ \text{trans}_x & \text{trans}_y & \text{pers}_2 \end{bmatrix}$$

The first two columns define standard affine transformations, including scaling, skewing, and translation. The third column (`pers_0`, `pers_1`, and `pers_2`) is typically used for perspective transformations, though in most affine transformations, it remains as `c(0, 0, 1)`.

See Also

- Matrix in skia_safe::matrix - Rust

Index