Package: nanosvgr (via r-universe)

May 15, 2026

Type Package

Title Read 'SVG' as Polygons

Version 0.1.0

Maintainer Mike Cheng <mikefc@coolbutuseless.com>

URL https://github.com/coolbutuseless/nanosvgr

BugReports https://github.com/coolbutuseless/nanosvgr/issues

Description Parse SVG to polygons, bezier curves, lines and associated graphical parameters. This enables the manipulation of the raw SVG data and custom plotting techniques.

License MIT + file LICENSE

Copyright The included 'nanosvg' code is Copyright (c) (c) 2013-14 Mikko Mononen memon@inside.org. See 'COPYRIGHTS' for LICENSE for included code.

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

LinkingTo colorfast

Depends R (>= 4.1.0), colorfast

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

Date/Publication 2026-04-17 22:11:20 UTC

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

RemoteRef HEAD

RemoteSha b95073c3694cd5101481a5451b99c106efaf4fda

Contents

`nsvg_add_points` *Refresh points data to an nsvg object*

Description

This is usually done as part of calling `nsvg_read()`, but can be called separately if the points data needs to be calculated at a finer or coarser level.

Usage

```
nsvg_add_points(nsvg, n = 20)
```

Arguments

`nsvg`	'nsvg' object created using `nsvg_read()`
`n`	number of points to use when converting each bezier to a polyline. Default: 20.

Value

'nsvg' object with modified 'points' column. This is a list-column with each element being a data.frame of computed points along this path

Examples

```
fn <- system.file("sailboat.svg", package = 'nanosvgr', mustWork = TRUE)
nsvg <- nsvg_read(fn)
# Use a very low resolution conversion. Only 3 points for each bezier.
nsvg <- nsvg_add_points(nsvg, n = 3)
```

nsvg_explode *Explode an nsvg object by shifting individual shapes*

Description

A simple demonstration of how we now have access to all the geometry

Usage

```
nsvg_explode(nsvg, scale = 2)
```

Arguments

nsvg	'nsvg' object created using `nsvg_read()`
scale	scale factor for explosion. Default: 2

Value

Modified 'nsvg' object with all coordinates scaled by the given factors

Examples

```
fn <- system.file("sailboat.svg", package = 'nanosvgr', mustWork = TRUE)
nsvg <- nsvg_read(fn) |> nsvg_scale(0.3)
grid::grid.newpage(); plot(nsvg)
nsvg <- nsvg_explode(nsvg, scale = 2)
grid::grid.newpage(); plot(nsvg)
```

nsvg_invert_closedness *Invert the closed-ness of each shape*

Description

Some SVGs will have path closedness incorrectly set. This is a helper function to invert the closedness of all paths. Try this if the plot output is empty.

Usage

```
nsvg_invert_closedness(nsvg)
```

Arguments

nsvg	'nsvg' object created using `nsvg_read()`

Value

Modifed 'nsvg' object with the 'closed' status of all paths inverted

Examples

```
fn <- system.file("shuttle.svg", package = 'nanosvgr', mustWork = TRUE)
nsvg <- nsvg_read(fn) |> nsvg_scale(0.3)
grid::grid.newpage(); plot(nsvg)
nsvg <- nsvg_invert_closedness(nsvg)
grid::grid.newpage(); plot(nsvg)
```

---

nsvg_read *Read an SVG as geometry*

Description

Read an SVG as geometry

Usage

```
nsvg_read(filename, n = 20, units = "px", dpi = 96)
```

Arguments

filename	SVG filename
n	number of points to use when converting each bezier to a polyline. Default: 20. Use "NA" to indicate that no conversion should be done.
units	units to use. Default 'px'. One of 'px', 'pt', 'pc' 'mm', 'cm', or 'in'
dpi	dots per inch. Default: 96

Value

An nsvg object which is a data.frame of SVG data. An SVG is made up of one-or-more *shapes*. Each *shape* contains one-or-more *paths*. Each *path* is made of one-or-more *cubic beziers*. Each shape has a set of graphical parameters

shape_idx [int] Shape index. Each SVG is defined as a number of shapes, with each shape having a number of *paths*.

fill [chr] fill color

stroke [chr] stroke color

alpha [dbl] opacity in range [0, 1]

lwd [dbl] line width

linejoin [chr] Line join style. 'bevel', 'mitre' or 'round'

lineend [chr] Line end style. 'round', 'butt', 'square'

`linemitre` [dbl] Line mitre limit

`linedash` [list] Raw list of line dash lengths for each shape

`fill_rule` [chr] 'evenodd' or 'winding'

`fill_type` [chr] 'flat', 'linear', 'radial', 'none', 'undef'

`gradient` [list] Radial or linear gradient information for this shape

`stroke_type` [chr] 'flat', 'linear', 'radial', 'none', 'undef'

`beziers` [list] A list of data.frames - one data.frame for each shape containing the coordinates of the bezier control points (Note: there are 4 control points for each cubic bezier).

- `path_idx` [int] Index of path within shape
- `bez_idx` [int] Index of bezier with path
- `closed` [lgl] Is the path closed?
- `x` [dbl] x coordinate of bezier control points
- `y` [dbl] y coordinate of bezier control points

`lty` [chr] Line type. Either 'solid' or a string of up to 8 characters (from c(1:9, "A":"F")) may be given, giving the length of line segments which are alternatively drawn and skipped. See `?graphics::par` for deails on Line Type Specification

`points` [list] A list of data.frames - one data.frame for each shape containing the polylines derived from the beziers

- `path_idx` [int] Index of path within shape
- `bez_idx` [int] Index of bezier with path
- `closed` [lgl] Is the path closed?
- `x` [dbl] x coordinate of polylines
- `y` [dbl] y coordinate of polylines

Examples

```
fn <- system.file("sailboat.svg", package = 'nanosvgr', mustWork = TRUE)
nsvg <- nsvg_read(fn)
head(nsvg[, 1:8])
grid::grid.newpage()
plot(nsvg)
# Manually change the fill color for each shape
nsvg$fill <- topo.colors(nrow(nsvg))
grid::grid.newpage()
plot(nsvg)
```

---

nsvg_scale *Scale nsvg object*

Description

Scale nsvg object

Usage

```
nsvg_scale(nsvg, scalex = 1, scaley = scalex)
```

Arguments

nsvg	'nsvg' object created using nsvg_read()
scalex, scaley	scale factors. Default: 1

Value

Modified 'nsvg' object with all coordinates scaled by the given factors

Examples

```
fn <- system.file("sailboat.svg", package = 'nanosvgr', mustWork = TRUE)
nsvg <- nsvg_read(fn)
grid::grid.newpage(); plot(nsvg)

nsvg <- nsvg_scale(nsvg, 0.3)
grid::grid.newpage(); plot(nsvg)
```

---

nsvg_to_grob *Conver nsvg object to grid graphics objects*

Description

Conver nsvg object to grid graphics objects

Usage

```
nsvg_to_grob(nsvg, n = 20, inverty = TRUE)
```

Arguments

nsvg	nsvg object
n	if beziers not yet flatted to points, will use this value
inverty	logical. Invert the y-axis? Default: TRUE

Value

A grid graphics grob

Examples

```
fn <- system.file("sailboat.svg", package = 'nanosvgr', mustWork = TRUE)
nsvg <- nsvg_read(fn)
grob <- nsvg_to_grob(nsvg)
grob
grid::grid.draw(grob)
```

`nsvg_unnest_beziers` *Unnest SVG bezier coordinates into a single long data.frame*

Description

Unnest SVG bezier coordinates into a single long data.frame

Usage

```
nsvg_unnest_beziers(nsvg, inverty = TRUE, gpars = FALSE)
```

Arguments

`nsvg`	'nsvg' object created using `nsvg_read()`
`inverty`	logical. Invert the y-axis? Default: TRUE
`gpars`	logical. Include graphical parameters in result? Default: FALSE

Value

data.frame with a row for each bezier control point.

Examples

```
fn <- system.file("sailboat.svg", package = 'nanosvgr', mustWork = TRUE)
nsvg <- nsvg_read(fn)
bezs <- nsvg_unnest_beziers(nsvg)
head(bezs)
```

nsvg_unnest_points *Unnest SVG polylines into a single long data.frame*

Description

Unnest SVG polylines into a single long data.frame

Usage

```
nsvg_unnest_points(nsvg, inverty = TRUE, gpars = FALSE)
```

Arguments

nsvg	'nsvg' object created using nsvg_read()
inverty	logical. Invert the y-axis? Default: TRUE
gpars	logical. Include graphical parameters in result? Default: FALSE

Value

data.frame with a row for each coordinate for each polyline calculated from the original beziers.

Examples

```
fn <- system.file("sailboat.svg", package = 'nanosvgr', mustWork = TRUE)
nsvg <- nsvg_read(fn)
points <- nsvg_unnest_points(nsvg)
head(points)
```

plot.nsvg *Plot SVG object*

Description

Plot SVG object

Usage

```
## S3 method for class 'nsvg'
plot(x, ...)
```

Arguments

x	nsvg object
...	ignored

Value

return nsvg object invisibly

Examples

```
fn <- system.file("sailboat.svg", package = 'nanosvgr', mustWork = TRUE)
nsvg <- nsvg_read(fn)
plot(nsvg)
```

Index