Package: textures (via r-universe)

September 14, 2024

Title Plot 3D Textures as 2D Graphics (Kinda)

Version 0.0.0.9023

Description Illustrate the use of texture mapping in rgl for depicting images in graphics or spatial coordinate systems, and mapped onto arbitrary shapes.

License GPL-3

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Imports rgl, scales

Depends R (>= 3.6.0)

Suggests testthat, anglr, knitr, png, rmarkdown

LinkingTo cpp11, quad

VignetteBuilder knitr

SystemRequirements C++11

Remotes hypertidy/quad

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

RemoteUrl https://github.com/hypertidy/textures

RemoteRef HEAD

RemoteSha 85912c68fb082c152ed3a64c9d0efa2bbb9918d4

Contents

break_mesh *un mesh*

Description

Break the topology of a mesh by expanding all vertices.

Usage

```
break_mesh(x)
```

Arguments

x	mesh3d, from e.g. quad()

Details

Details ... rgl is inherently *topological*, but we can have primitives that are geometrically independent. (One day I'll find a way to talk about this that's not garble.)

Value

mesh3d

Examples

```
(mesh <- quad(depth = 3))
## same number of primitives, more vertices (every coordinate)
break_mesh(mesh)
```

ga_topo *Topographic image*

Description

Image of Australia as a map, its extent, and map projection.

Usage

```
ga_topo
```

Format

A list with an array, a numeric vector, and a character vector:

img image array with dimension 921,1025,3 - three slices Red, Green, Blue

extent the geographic extent of the array, in metres

crs the map projection of the geographic extent of img

Details

(It's web Mercator, aka 'EPSG:3857'. We've kept the proj string because it's the easiest to use atm - May 2020.)

Provenance

The image is named 'Australian Topographic Base Map (Web Mercator)' and is from the following Geoscience Australia Web Map Tile Service (WMTS): http://gaservices.ga.gov.au/site_7/rest/services/Topographic_Base_Map_WM/MapServer.

Code to obtain the image is in 'data-raw/ga_topo.R' at https://github.com/hypertidy/textures using the wmts package https://github.com/mdsumner/wmts.

---

plot.mesh3d *Plot (2D) for mesh3d*

Description

Plot (2D) for mesh3d

Usage

```
## S3 method for class 'mesh3d'
plot(
  x,
  ...,
  asp = 1,
  add = FALSE,
  axes = TRUE,
  border = "black",
  col = NA,
  alpha = 1,
  lwd = 1,
  lty = 1
)
```

Arguments

x	mesh3d object (with any or all of quads, triangles, segments)

Value

nothing, called for side effect of graphics

png_plot3d *Plot a PNG bitmap in 3D*

Description

Plot a PNG bitmap in 3D

Usage

```
png_plot3d(pngfile, dim = c(1, 1))
```

Arguments

pngfile	path to a PNG format image file
dim	specify dimensions of quad grid see quad()

Value

returns a mesh3d with 1 quad and the image file textured to it, as a side effect creates a 3D interactive plot

Examples

```
file <- system.file("extdata/Rlogo.png", package = "textures")
png_plot3d(file)
```

quad *Quad canvas*

Description

Create a simple quad mesh3d object

Usage

```
quad(dimension = c(1L, 1L), extent = NULL, ydown = FALSE, ...)

quad_texture(dimension = c(1L, 1L), extent = NULL, ydown = FALSE, texture = "")

segs(dimension = c(1L, 1L), extent = NULL, ydown = FALSE, ...)
```

Arguments

dimension	dimensions of mesh (using matrix() and image() orientation)
extent	optional extent of mesh xmin, xmax, ymin, ymax
ydown	should y-coordinate be counted from top (default FALSE)
...	used only to warn about old usage
texture	file path to PNG image (may not exist)

Details

Use quad() to create a mesh3d object with quad indexes to the vertices, this is defined in the rgl package by qmesh3d() and has elements vb (the homogeneous coordinates 4xn) and ib (the quad index 4xn).

Use seg() to create a mesh3d object with segment indexes, exactly analogous to the mesh created by quad() just only containing the quad edges/segments - note that segments are unique.

The meshColor is currently hardcoded as 'vertices'.

Use quad_texture() to create a mesh3d object additionally with texcoords and texture properties.

Value

mesh3d with quads and material texture settings as per inputs

Deprecation note

Note that an early version used arguments 'depth' (to control rgl::subdivision3d()), 'tex' to indicate that texture should be included, 'texfile' a link to the texture file path, and 'unmesh' to remove topology by expanding the vertices . Please now use quad_texture() for textures, and dimension argument (length 1 or 2), and break_mesh().

Examples

```
qm <- quad()
## orientation is low to high, x then y
qm <- quad(dim(volcano))
scl <- function(x) (x - min(x, na.rm = TRUE))/diff(range(x, na.rm = TRUE))
qm$meshColor <- "faces"
qm$material$color <- hcl.colors(12, "YlOrRd", rev = TRUE)[scl(volcano) * 11 + 1]
rgl::plot3d(qm)
```

`set_scene` *rgl defaults*

Description

Quick defaults for rgl static plot

Usage

```
set_scene(
  interactive = FALSE,
  zoom = 0.5,
  phi = 0,
  theta = 0,
  light_phi = -45,
  light_theta = 0
)
```

Arguments

`interactive`	mouse interactive plot, default `FALSE`
`zoom`	zoom for rgl::view3d default `0.5` which is closer than rgl 1
`phi`, `theta`	polar coordinates, passed to `rgl::view3d()`
`light_phi`, `light_theta`	polar coordinates, passed to `rgl::light3d()` as `phi` and `theta` respectively

Details

This function sets the size of the window to 1024x1024, sets the view position at directly vertical `phi = 0, theta = 0`, makes the view non-interactive (zoom is enabled, but no pivot or pan). It turns off the lights and puts a new light in front of the viewer (to avoid shiny glare), sets the aspect ratio to 'iso' ("fill the box"), and attempts to 'bringtotop', but I think that has to happen interactively with `rgl::rgl.bringtotop()` (especially for animating or snapshotting scenes to file).

`phi` and `theta` use `0` and `0` respectively, phi is different from rgl's default in order to look straight down on the quad (along the z axis)

`light_phi` and `light_theta` use `-45` and `0` respectively, phi is different from rg's default to put the light source forwards (y+) from the viewer when looking straight down

Value

nothing

Examples

```
## see README and in-dev examples in rough-examples.R
rgl::plot3d(rnorm(10), rnorm(10), rnorm(1)); set_scene()
```

Index