Package ‘rayshader’

April 18, 2020

Type Package
Title Create Maps and Visualize Data in 2D and 3D
Version 0.15.1
Author Tyler Morgan-Wall
Maintainer Tyler Morgan-Wall <tylermw@gmail.com>
Description Uses a combination of raytracing and multiple hill shading methods to produce 2D and 3D data visualizations and maps. Includes water detection and layering functions, programmable color palette generation, several built-in textures for hill shading, 2D and 3D plotting options, a built-in path tracer, 'Wavefront' OBJ file export, and the ability to save 3D visualizations to a 3D printable format.
License GPL-3
LazyData true
Depends R (>= 3.0.2)
Imports doParallel, foreach, Rcpp, progress, raster, scales, png, magrittr, rgl, grDevices, grid, utils, methods
Suggests reshape2, viridis, av, magick, ggplot2, sf, rayrender, rayimage
LinkingTo Rcpp, progress, RcppArmadillo
RoxygenNote 7.0.2
URL https://github.com/tylermorganwall/rayshader
BugReports https://github.com/tylermorganwall/rayshader/issues
NeedsCompilation yes
Repository CRAN
Date/Publication 2020-04-18 21:20:02 UTC

R topics documented:

  add_overlay .............................................................. 2
  add_shadow .............................................................. 4
add_overlay

Description

Overlays an image (with a transparency layer) on the current map.

Usage

```r
add_overlay(
  hillshade,
  overlay,
  alphacolor = NULL,
  alphalayer = 1,
  alphamethod = "max",
  gamma_correction = TRUE
)
```
**Arguments**

- **hillshade** A three-dimensional RGB array or 2D matrix of shadow intensities.
- **overlay** A three or four dimensional RGB array, where the 4th dimension represents the alpha (transparency) channel. If the array is 3D, ‘alphacolor’ should also be passed to indicate transparent regions.
- **alphacolor** Default ‘NULL’. If ‘overlay’ is a 3-layer array, this argument tells which color is interpreted as completely transparent.
- **alphalayer** Default ‘1’. Defines minimum transparency of layer. If transparency already exists in ‘overlay’, the way ‘add_overlay’ combines the two is determined in argument ‘alphamethod’.
- **alphamethod** Default ‘max’. Method for dealing with pre-existing transparency with ‘layeralpha’. If ‘max’, converts all alpha levels higher than ‘layeralpha’ to the value set in ‘layeralpha’. If ‘multiply’, multiples all pre-existing alpha values with ‘layeralpha’. If ‘none’, keeps existing transparencies and only changes opaque entries.
- **gamma_correction** Default ‘TRUE’. Controls gamma correction when adding colors. Default exponent of 2.2.

**Value**

Hillshade with overlay.

**Examples**

```r
# Here, we overlay a base R elevation plot with our raytraced shadow layer:

fliplr = function(x) {
  x[,ncol(x):1]
}
## Not run:
tempfilename = tempfile()
old.par = par(no.readonly = TRUE)
on.exit(par(old.par))
png(tempfilename,width = 401,height=401)
par(mar = c(0,0,0,0))
raster::image(fliplr(montereybay),axes = FALSE,col = rev(terrain.colors(1000)))
dev.off()
tempmap = png::readPNG(tempfilename)
## End(Not run)

## Not run:
montereybay %>%
  ray_shade(zscale=50,maxsearch = 500,anglebreaks = seq(20,30,0.1)) %>%
  add_overlay(tempmap,alphalayer = 0.5) %>%
  plot_map()
## End(Not run)
```
#Combining base R plotting with rayshader's spherical color mapping and raytracing:
## Not run:
montereybay %>%
sphere_shade() %>%
add_overlay(tempmap, alphalayer = 0.4) %>%
add_shadow(ray_shade(montereybay, zscale = 50, maxsearch = 500)) %>%
plot_map()
## End(Not run)

---

add_shadow Add Shadow

## Description

Multiplies a texture array or shadow map by a shadow map.

## Usage

```r
add_shadow(hillshade, shadowmap, max_darken = 0.7)
```

## Arguments

- `hillshade` A three-dimensional RGB array or 2D matrix of shadow intensities.
- `shadowmap` A matrix that indicates the intensity of the shadow at that point. 0 is full darkness, 1 is full light.
- `max_darken` Default 0.7. The lower limit for how much the image will be darkened. 0 is completely black, 1 means the shadow map will have no effect.

## Value

Shaded texture map.

## Examples

```r
#First we plot the sphere_shade() hillshade of `montereybay` with no shadows

montereybay %>%
sphere_shade(colorintensity = 0.5) %>%
plot_map()

#Raytrace the `montereybay` elevation map and add that shadow to the output of sphere_shade()

montereybay %>%
sphere_shade(colorintensity = 0.5) %>%
add_shadow(ray_shade(montereybay, sunaltitude = 20, zscale = 50), max_darken = 0.3) %>%
```
add_water

plot_map()

#Increase the intensity of the shadow map with the max_darken argument.

montereybay %>%
sphere_shade(colorintensity=0.5) %>%
add_shadow(ray_shade(montereybay,sunaltitude=20,zscale=50),max_darken=0.1) %>%
plot_map()

#Decrease the intensity of the shadow map.

montereybay %>%
sphere_shade(colorintensity=0.5) %>%
add_shadow(ray_shade(montereybay,sunaltitude=20,zscale=50),max_darken=0.7) %>%
plot_map()

description

add_water

Description

Adds a layer of water to a map.

Usage

add_water(hillshade, watermap, color = "imhof1")

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hillshade</td>
<td>A three-dimensional RGB array.</td>
</tr>
<tr>
<td>watermap</td>
<td>Matrix indicating whether water was detected at that point. 1 indicates water, 0 indicates no water.</td>
</tr>
<tr>
<td>color</td>
<td>Default 'imhof1'. The water fill color. A hexcode or recognized color string. Also includes built-in colors to match the palettes included in sphere_shade: ('imhof1', 'imhof2', 'imhof3', 'imhof4', 'desert', 'bw', and 'unicorn').</td>
</tr>
</tbody>
</table>

Examples

#Here we even out a portion of the volcano dataset to simulate water:

island_volcano = volcano

island_volcano[island_volcano < mean(island_volcano)] = mean(island_volcano)

#Setting a minimum area avoids classifying small flat areas as water:

island_volcano %>%
sphere_shade(texture="imhof3") %>%
add_water(detect_water(island_volcano, min_area = 400),color="imhof3") %>%
plot_map()

# We'll do the same thing with the Monterey Bay dataset to fill in the ocean:

montbay_water = montereybay
montbay_water[montbay_water < 0] = 0

montereybay %>%
  sphere_shade(texture="imhof4") %>%
  add_water(detect_water(montbay_water),color="imhof4") %>%
  plot_map()

ambient_shade

### Calculate Ambient Occlusion Map

**Description**

Calculates Ambient Occlusion Shadow Map

**Usage**

```r
ambient_shade(
  heightmap,
  anglebreaks = 90 * cospi(seq(5, 85, by = 5)/180),
  sunbreaks = 24,
  maxsearch = 30,
  multicore = FALSE,
  zscale = 1,
  cache_mask = NULL,
  shadow_cache = NULL,
  progbar = interactive(),
  ...
)
```

**Arguments**

- `heightmap`: A two-dimensional matrix, where each entry in the matrix is the elevation at that point. All points are assumed to be evenly spaced.
- `anglebreaks`: Default `90*cospi(seq(5, 85, by =5)/180)`. The angle(s), in degrees, as measured from the horizon from which the light originates.
- `sunbreaks`: Default `24`. Number of rays to be sent out in a circle, evenly spaced, around the point being tested.
- `maxsearch`: Default `30`. The maximum horizontal distance that the system should propagate rays to check for surface intersections.
- `multicore`: Default `FALSE`. If `TRUE`, multiple cores will be used to compute the shadow matrix. By default, this uses all cores available, unless the user has set `options("cores")` in which the multicore option will only use that many cores.
**calculate_normal**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>zsclae</td>
<td>Default 1. The ratio between the x and y spacing (which are assumed to be equal) and the z axis.</td>
</tr>
<tr>
<td>cache_mask</td>
<td>Default ‘NULL’. A matrix of 1 and 0s, indicating which points on which the raytracer will operate.</td>
</tr>
<tr>
<td>shadow_cache</td>
<td>Default ‘NULL’. The shadow matrix to be updated at the points defined by the argument ‘cache_mask’.</td>
</tr>
<tr>
<td>progbar</td>
<td>Default ‘TRUE’ if interactive, ‘FALSE’ otherwise. If ‘FALSE’, turns off progress bar.</td>
</tr>
<tr>
<td>...</td>
<td>Additional arguments to pass to the ‘makeCluster’ function when ‘multicore=TRUE’.</td>
</tr>
</tbody>
</table>

**Value**

Shaded texture map.

**Examples**

```
#Here we produce a ambient occlusion map of the `montereybay` elevation map.
## Not run:
amb = ambient_shade(heightmap = montereybay)
plot_map(amb)

## End(Not run)
```

```
#We can increase the distance to look for surface intersections `maxsearch`
#and the density of rays sent out around the point `sunbreaks`.
## Not run:
amb = ambient_shade(heightmap = montereybay,sunbreaks=24,maxsearch=50)
plot_map(amb)

## End(Not run)
```

---

**Description**

Calculates the normal unit vector for every point on the grid.

**Usage**

```
calculate_normal(heightmap, zsclae = 1, progbar = FALSE)
```

**Arguments**

- `heightmap` A two-dimensional matrix, where each entry in the matrix is the elevation at that point. All points are assumed to be evenly spaced.
- `zsclae` Default 1.
- `progbar` Default ‘FALSE’. If ‘TRUE’, turns on progress bar.
create_texture  

Description  

Creates a texture map based on 5 user-supplied colors.

Usage  

create_texture(  
  lightcolor,  
  shadowcolor,  
  leftcolor,  
  rightcolor,  
  centercolor,  
  cornercolors = NULL  
)

Arguments  

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lightcolor</td>
<td>The main highlight color. Corresponds to the top center of the texture map.</td>
</tr>
<tr>
<td>shadowcolor</td>
<td>The main shadow color. Corresponds to the bottom center of the texture map.</td>
</tr>
<tr>
<td>This color represents slopes directed directly opposite to the main highlight color.</td>
<td></td>
</tr>
<tr>
<td>leftcolor</td>
<td>The left fill color. Corresponds to the left center of the texture map. This color represents slopes directed 90 degrees to the left of the main highlight color.</td>
</tr>
<tr>
<td>rightcolor</td>
<td>The right fill color. Corresponds to the right center of the texture map. This color represents slopes directed 90 degrees to the right of the main highlight color.</td>
</tr>
<tr>
<td>centercolor</td>
<td>The center color. Corresponds to the center of the texture map. This color represents flat areas.</td>
</tr>
<tr>
<td>cornercolors</td>
<td>Default ‘NULL’. The colors at the corners, in this order: NW, NE, SW, SE. If this vector isn’t present (or all corners are specified), the mid-points will just be interpolated from the main colors.</td>
</tr>
</tbody>
</table>
**detect_water**

**Examples**

#Here is the `imhof\` palette:
create_texture("#fff673","#55967a","#8fb28a","#55967a","#cfe0a9") %>%
plot_map()

#Here is the `unicorn` palette:
create_texture("red","green","blue","yellow","white") %>%
plot_map()

**detect_water**

**Detect water**

**Description**

Detects bodies of water (of a user-defined minimum size) within an elevation matrix.

**Usage**

```r
detect_water(
  heightmap,
  zscale = 1,
  cutoff = 0.999,
  min_area = length(heightmap)/400,
  max_height = NULL,
  normalvectors = NULL,
  keep_groups = FALSE,
  progbar = FALSE
)
```

**Arguments**

- **heightmap**: A two-dimensional matrix, where each entry in the matrix is the elevation at that point. All grid points are assumed to be evenly spaced.
- **zscale**: Default ‘1’. The ratio between the x and y spacing (which are assumed to be equal) and the z axis. For example, if the elevation levels are in units of 1 meter and the grid values are separated by 10 meters, ‘zscale’ would be 10.
- **cutoff**: Default ‘0.999’. The lower limit of the z-component of the unit normal vector to be classified as water.
- **min_area**: Default length(heightmap)/400. Minimum area (in units of the height matrix x and y spacing) to be considered a body of water.
- **max_height**: Default ‘NULL’. If passed, this number will specify the maximum height a point can be considered to be water.
- **normalvectors**: Default ‘NULL’. Pre-computed array of normal vectors from the `calculate_normal` function. Supplying this will speed up water detection.
- **keep_groups**: Default ‘FALSE’. If ‘TRUE’, the matrix returned will retain the numbered grouping information.
- **progbar**: Default ‘FALSE’. If ‘TRUE’, turns on progress bar.
height_shade

Value

Matrix indicating whether water was detected at that point. 1 indicates water, 0 indicates no water.

Examples

library(magrittr)
# Here we even out a portion of the volcano dataset to simulate water:
island_volcano = volcano
island_volcano[island_volcano < mean(island_volcano)] = mean(island_volcano)

# Setting a minimum area avoids classifying small flat areas as water:
island_volcano %>%
sphere_shade(texture="imhof3") %>%
add_water(detect_water(island_volcano, min_area = 400), color="imhof3") %>%
plot_map()

height_shade

Calculate Terrain Color Map

Description

Calculates a color for each point on the surface using a direct elevation-to-color mapping.

Usage

height_shade(heightmap, texture = grDevices::terrain.colors(256))

Arguments

heightmap A two-dimensional matrix, where each entry in the matrix is the elevation at that point.
texture Default ‘terrain.colors(256)’. A color palette for the plot.

Value

RGB array of hillshaded texture mappings.

Examples

# Create a direct mapping of elevation to color:
montereybay %>%
height_shade() %>%
plot_map()

# Add a shadow:
montereybay %>%
height_shade() %>%
add_shadow(ray_shade(montereybay, zscale=50), 0.3) %>>%
# Change the palette:

```r
montereybay %>%
  height_shade(texture = topo.colors(256)) %>%
  add_shadow(ray_shade(montereybay, zscale=50), 0.3) %>%
  plot_map()
```

# Really change the palette:

```r
montereybay %>%
  height_shade(texture = rainbow(256)) %>%
  add_shadow(ray_shade(montereybay, zscale=50), 0.3) %>%
  plot_map()
```

---

**lamb_shade**  
*Calculate Lambert Shading Map*

**Description**

Calculates local shadow map for a elevation matrix by calculating the dot product between light direction and the surface normal vector at that point. Each point’s intensity is proportional to the cosine of the normal vector.

**Usage**

```r
lamb_shade(
  heightmap,
  sunaltitude = 45,
  sunangle = 315,
  zscale = 1,
  zero_negative = TRUE
)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>heightmap</td>
<td>A two-dimensional matrix, where each entry in the matrix is the elevation at that point. All points are assumed to be evenly spaced.</td>
</tr>
<tr>
<td>sunaltitude</td>
<td>Default <code>45</code>°. The azimuth angle as measured from the horizon from which the light originates.</td>
</tr>
<tr>
<td>sunangle</td>
<td>Default <code>315</code>° (NW). The angle around the matrix from which the light originates.</td>
</tr>
<tr>
<td>zscale</td>
<td>Default <code>1</code>. The ratio between the x and y spacing (which are assumed to be equal) and the z axis.</td>
</tr>
</tbody>
</table>
zero_negative Default 'TRUE'. Zeros out all values below 0 (corresponding to surfaces facing away from the light source).

Value

Matrix of light intensities at each point.

Examples

#Here we produce a light intensity map of the `volcano` elevation map.
volcanointensity = lamb_shade(heightmap = volcano,
   sunaltitude = 60,
   sunangle = 25,
   zscale = 1)

plot_map(volcanointensity)

montereybay Monterey Bay combined topographic and bathymetric elevation matrix.

Description

This dataset is a downsampled version of a combined topographic and bathymetric elevation matrix representing the Monterey Bay, CA region. Original data from from the NOAA National Map website.

Usage

montereybay

Format

A matrix with 401 rows and 401 columns. Elevation is in meters, and the spacing between each coordinate is 200 meters (zscale = 200). Water level is 0.

Source

https://maps.ngdc.noaa.gov/viewers/bathymetry/?layers=dem
**plot_3d**

**Plot 3D**

**Description**

Displays the shaded map in 3D with the ‘rgl’ package.

**Usage**

```r
plot_3d(
  hillshade,
  heightmap,
  zscale = 1,
  baseshape = "rectangle",
  solid = TRUE,
  soliddepth = "auto",
  solidcolor = "grey20",
  solidlinecolor = "grey30",
  shadow = TRUE,
  shadowdepth = "auto",
  shadowcolor = "grey50",
  shadowwidth = "auto",
  water = FALSE,
  waterdepth = 0,
  watercolor = "lightblue",
  wateralpha = 0.5,
  waterlinecolor = NULL,
  waterlinealpha = 1,
  linewidth = 2,
  lineantialias = FALSE,
  theta = 45,
  phi = 45,
  fov = 0,
  zoom = 1,
  background = "white",
  calculate_normals = TRUE,
  litbase = FALSE,
  windowsize = 600,
  ...
)
```

**Arguments**

- **hillshade**  
  Hillshade/image to be added to 3D surface map.

- **heightmap**  
  A two-dimensional matrix, where each entry in the matrix is the elevation at that point. All points are assumed to be evenly spaced.
zscale
Default ‘1’. The ratio between the x and y spacing (which are assumed to be equal) and the z axis. For example, if the elevation levels are in units of 1 meter and the grid values are separated by 10 meters, ‘zscale’ would be 10. Adjust the zscale down to exaggerate elevation features.

baseshape
Default ‘rectangle’. Shape of the base. Options are c("rectangle", "circle", "hex").
solid
Default ‘TRUE’. If ‘FALSE’, just the surface is rendered.
soliddepth
Default ‘auto’, which sets it to the lowest elevation in the matrix minus one unit (scaled by zscale). Depth of the solid base.
solidcolor
Default ‘grey20’. Base color.
solidlinecolor
Default ‘grey30’. Base edge line color.
shadow
Default ‘TRUE’. If ‘FALSE’, no shadow is rendered.
shadowdepth
Default ‘auto’, which sets it to ‘soliddepth - soliddepth/10’. Depth of the shadow layer.
shadowcolor
Default ‘grey50’. Color of the shadow.
shadowwidth
Default ‘auto’, which sizes it to 1/10th the smallest dimension of ‘heightmap’. Width of the shadow in units of the matrix.
water
Default ‘FALSE’. If ‘TRUE’, a water layer is rendered.
waterdepth
Default ‘0’. Water level.
watercolor
Default ‘lightblue’. Color of the water.
wateralpha
Default ‘0.5’. Water transparency.
waterlinecolor
Default ‘NULL’. Color of the lines around the edges of the water layer.
waterlinealpha
Default ‘1’. Water line transparency.
linewidth
Default ‘2’. Width of the edge lines in the scene.
lineantialias
Default ‘FALSE’. Whether to anti-alias the lines in the scene.
theta
Default ‘45’. Rotation around z-axis.
phi
Default ‘45’. Azimuth angle.
fov
Default ‘0’–isometric. Field-of-view angle.
zoom
Default ‘1’. Zoom factor.
background
Default ‘grey10’. Color of the background.
calculate_normals
Default ‘TRUE’. If ‘FALSE’, will not calculate per-vertex normals. Can also pass the output of the function ‘calculate_normals’ to use pre-computed normals.
litbase
Default ‘FALSE’. If ‘TRUE’, the base will be a glossy material lit using the ‘lit = TRUE’ parameter in ‘rgl’. If calling ‘render_highquality()’, set this to ‘TRUE’ to make sure rgl exports the sizes of the model.
windowsize
Default ‘600’. Position, width, and height of the ‘rgl’ device displaying the plot. If a single number, viewport will be a square and located in upper left corner. If two numbers, (e.g. ‘c(600,800)’), user will specify width and height separately. If four numbers (e.g. ‘c(200,0,600,800)’), the first two coordinates specify the location of the x-y coordinates of the bottom-left corner of the viewport on the
screen, and the next two (or one, if square) specify the window size. NOTE: The absolute positioning of the window does not currently work on macOS (tested on Mojave), but the size can still be specified.

Examples

# Plotting a spherical texture map of the built-in 'montereybay' dataset.

```r
montereybay %>%
  sphere_shade(texture="desert") %>%
  plot_3d(montereybay,zscale=50)
render_snapshot(clear = TRUE)
```

# With a water layer

```r
montereybay %>%
  sphere_shade(texture="imhof2") %>%
  plot_3d(montereybay, zscale=50, water = TRUE, watercolor="imhof2",
    waterlinecolor="white", waterlinealpha=0.5)
render_snapshot(clear = TRUE)
```

# We can also change the base by setting "baseshape" to "hex" or "circle"

```r
montereybay %>%
  sphere_shade(texture="imhof1") %>%
  plot_3d(montereybay, zscale=50, water = TRUE, watercolor="imhof1",
    theta=-45, zoom=0.7,
    waterlinecolor="white", waterlinealpha=0.5,baseshape="circle")
render_snapshot(clear = TRUE)
```

# Or we can carve out the region of interest ourselves, by setting those entries to NA
to the elevation map passed into 'plot_3d'

# Here, we only include the deep bathymetry data by setting all points greater than -10
# in the copied elevation matrix to NA.

```r
mb_water = montereybay
mb_water[mb_water > -10] = NA
```

```r
montereybay %>%
  sphere_shade(texture="imhof1") %>%
  plot_3d(montereybay, zscale=50, water = TRUE, watercolor="imhof1",
    theta=-45, zoom=0.7,
    waterlinecolor="white", waterlinealpha=0.5,baseshape="hex")
render_snapshot(clear = TRUE)
```
Description

Plots a ggplot2 object in 3D by mapping the color or fill aesthetic to elevation. Currently, this function does not transform lines mapped to color into 3D.

If there are multiple legends/guides due to multiple aesthetics being mapped (e.g. color and shape), the package author recommends that the user pass the order of the guides manually using the ggplot2 function `guides()`. Otherwise, the order may change when processing the ggplot2 object and result in a mismatch between the 3D mapping and the underlying plot.

Using the shape aesthetic with more than three groups is not recommended, unless the user passes in custom, solid shapes. By default in ggplot2, only the first three shapes are solid, which is a requirement to be projected into 3D.

Usage

```r
plot_gg(
  ggobj,
  width = 3,
  height = 3,
  height_aes = NULL,
  invert = FALSE,
  shadow_intensity = 0.5,
  units = c("in", "cm", "mm"),
  scale = 150,
  pointcontract = 0.7,
  offset_edges = FALSE,
  preview = FALSE,
  raytrace = TRUE,
  sunangle = 315,
  anglebreaks = seq(30, 40, 0.1),
  multicore = FALSE,
  lambert = TRUE,
  reduce_size = NULL,
  save_height_matrix = FALSE,
  save_shadow_matrix = FALSE,
  saved_shadow_matrix = NULL,
  ...
)
```
Arguments

**ggobj**
- ggplot object to projected into 3D.

**width**
- Default ‘3’. Width of ggplot, in ‘units’.

**height**
- Default ‘3’. Height of ggplot, in ‘units’.

**height_aes**
- Default ‘NULL’. Whether the ‘fill’ or ‘color’ aesthetic should be used for height values, which the user can specify by passing either ‘fill’ or ‘color’ to this argument. Automatically detected. If both ‘fill’ and ‘color’ aesthetics are present, then ‘fill’ is default.

**invert**
- Default ‘FALSE’. If ‘TRUE’, the height mapping is inverted.

**shadow_intensity**
- Default ‘0.5’. The intensity of the calculated shadows.

**units**
- Default ‘in’. One of c("in", "cm", "mm").

**scale**
- Default ‘150’. Multiplier for vertical scaling: a higher number increases the height of the 3D transformation.

**pointcontract**
- Default ‘0.7’. This multiplies the size of the points and shrinks them around their center in the 3D surface mapping. Decrease this to reduce color bleed on edges, and set to ‘1’ to turn off entirely. Note: If ‘size’ is passed as an aesthetic to the same geom that is being mapped to elevation, this scaling will not be applied. If ‘alpha’ varies on the variable being mapped, you may want to set this to ‘1’, since the points now have a non-zero width stroke outline (however, mapping ‘alpha’ in the same variable you are projecting to height is probably not a good choice, as the ‘alpha’ variable is ignored when performing the 3D projection).

**offset_edges**
- Default ‘FALSE’. If ‘TRUE’, inserts a small amount of space between polygons for "geom_sf", "geom_tile", "geom_hex", and "geom_polygon" layers. If you pass in a number, the space between polygons will be a line of that width. Note: this feature may end up removing thin polygons from the plot entirely—use with care.

**preview**
- Default ‘FALSE’. If ‘TRUE’, the raytraced 2D ggplot will be displayed on the current device.

**raytrace**
- Default ‘FALSE’. Whether to add a raytraced layer.

**sunangle**
- Default ‘315’ (NW). If raytracing, the angle (in degrees) around the matrix from which the light originates.

**anglebreaks**
- Default ‘seq(30,40,0.1)’. The azimuth angle(s), in degrees, as measured from the horizon from which the light originates.

**multicore**
- Default ‘FALSE’. If raytracing and ‘TRUE’, multiple cores will be used to compute the shadow matrix. By default, this uses all cores available, unless the user has set ‘options("cores")’ in which the multicore option will only use that many cores.

**lambert**
- Default ‘TRUE’. If raytracing, changes the intensity of the light at each point based proportional to the dot product of the ray direction and the surface normal at that point. Zeros out all values directed away from the ray.
reduce_size Default ‘NULL’. A number between ‘0’ and ‘1’ that specifies how much to reduce the resolution of the plot, for faster plotting. By default, this just decreases the size of height map, not the image. If you wish the image to be reduced in resolution as well, pass a numeric vector of size 2.

save_height_matrix Default ‘FALSE’. If ‘TRUE’, the function will return the height matrix used for the ggplot.

save_shadow_matrix Default ‘FALSE’. If ‘TRUE’, the function will return the shadow matrix for use in future updates via the ‘shadow_cache’ argument passed to ‘ray_shade’.

saved_shadow_matrix Default ‘NULL’. A cached shadow matrix (saved by the a previous invocation of ‘plot_gg(..., save_shadow_matrix=TRUE)’ to use instead of raytracing a shadow map each time.

... Additional arguments to be passed to ‘plot_3d()’.

Value

Opens a 3D plot in rgl.

Examples

library(ggplot2)
library(viridis)

ggdiamonds = ggplot(diamonds, aes(x, depth)) +
  stat_density_2d(aes(fill = stat(nlevel)), geom = “polygon”, n = 100, bins = 10, contour = TRUE) +
  facet_wrap(clarity~.) +
  scale_fill_viridis_c(option = “A”)

plot_gg(ggdiamonds,multicore=TRUE,width=5,height=5,scale=250,windowsize=c(1400,866),
  zoom = 0.55, phi = 30)
render_snapshot()

#Change the camera angle and take a snapshot:
render_camera(zoom=0.5,theta=-30,phi=30)
render_snapshot(clear = TRUE)

#Contours and other lines will automatically be ignored. Here is the volcano dataset:

ggvolcano = volcano %>%
  reshape2::melt() %>%
  ggplot() +
  geom_tile(aes(x=Var1,y=Var2,fill=value)) +
  geom_contour(aes(x=Var1,y=Var2,z=value),color=”black”) +
  scale_x_continuous(“X”,expand = c(0,0)) +
  scale_y_continuous(“Y”,expand = c(0,0)) +
  ...
Here, we will create a 3D plot of the mtcars dataset. This automatically detects that the user used the `color` aesthetic instead of the `fill`.

```r
mtplot = ggplot(mtcars) +
  geom_point(aes(x=mpg,y=disp,color=cyl)) +
  scale_color_continuous(limits=c(0,8))
```

# Preview how the plot will look by setting `preview = TRUE`: We also adjust the angle of the light.

```r
plot_gg(mtplot, width=3.5, sunangle=225, preview = TRUE)
```

```r
plot_gg(mtplot, width=3.5, multicore = TRUE, windowsize = c(1400,866), sunangle=225,
  zoom = 0.60, phi = 30, theta = 45)
```

# Now let's plot a density plot in 3D.

```r
mtplot_density = ggplot(mtcars) +
  stat_density_2d(aes(x=mpg,y=disp, fill=..density..), geom = "raster", contour = FALSE) +
  scale_x_continuous(expand=c(0,0)) +
  scale_y_continuous(expand=c(0,0)) +
  scale_fill_gradient(low="pink", high="red")
```

```r
mtplot_density
```

```r
plot_gg(mtplot_density, width = 4,zoom = 0.60, theta = -45, phi = 30,
  windowsize = c(1400,866))
```

# This also works facetted.

```r
mtplot_density_facet = mtplot_density + facet_wrap(~cyl)
```

# Preview this plot in 2D:

```r
plot_gg(mtplot_density_facet, preview = TRUE)
```

```r
plot_gg(mtplot_density_facet, windowsize=c(1400,866),
  zoom = 0.55, theta = -10, phi = 25)
```
plot_map

Description

Displays the map in the current device.

Usage

plot_map(hillshade, rotate = 0, keep_user_par = FALSE, ...)

Arguments

hillshade  Hillshade to be plotted.
keep_user_par Default ‘TRUE’. Whether to keep the user’s ‘par()’ settings. Set to ‘FALSE’ if you want to set up a multi-pane plot (e.g. set `par(mfrow)`).
...         Additional arguments to pass to the ‘raster::plotRGB’ function that displays the map.

Examples

#Plotting a spherical texture map of the volcano dataset.
plot_map(sphere_shade(volcano))
raster_to_matrix

**Description**

Turns a raster into a matrix suitable for rayshader.

**Usage**

```r
raster_to_matrix(raster, verbose = interactive())
```

**Arguments**

- `raster`: The input raster. Either a RasterLayer object, or a filename.
- `verbose`: Default `interactive()`. Will print dimensions of the resulting matrix.

**Examples**

```r
#Save montereybay as a raster and open using the filename.

temp_raster_filename = paste0(tempfile(),".tif")
raster::writeRaster(raster::raster(t(montereybay)),temp_raster_filename)
elmat = raster_to_matrix(temp_raster_filename)
elmat %>%
sphere_shade() %>
plot_map()
```

---

ray_shade

**Calculate Raytraced Shadow Map**

**Description**

Calculates shadow map for a elevation matrix by propogating rays from each matrix point to the light source(s), lowering the brightness at each point for each ray that intersects the surface.

**Usage**

```r
ray_shade(
  heightmap,
  sunaltitude = 45,
  sunangle = 315,
  maxsearch = NULL,
  lambert = TRUE,
  zscale = 1,
)```
Arguments

heightmap A two-dimensional matrix, where each entry in the matrix is the elevation at that point. All points are assumed to be evenly spaced.

sunaltitude Default ‘45’. The angle, in degrees (as measured from the horizon) from which the light originates. The width of the light is centered on this value and has an angular extent of 0.533 degrees, which is the angular extent of the sun. Use the ‘anglebreaks’ argument to create a softer (wider) light. This has a hard minimum/maximum of 0/90 degrees.

sunangle Default ‘315’ (NW). The angle, in degrees, around the matrix from which the light originates. Zero degrees is North, increasing clockwise.

maxsearch Defaults to the longest possible shadow given the ‘sunaltitude’ and ‘heightmap’. Otherwise, this argument specifies the maximum distance that the system should propagate rays to check.

lambert Default ‘TRUE’. Changes the intensity of the light at each point based proportional to the dot product of the ray direction and the surface normal at that point. Zeros out all values directed away from the ray.

zscale Default ‘1’. The ratio between the x and y spacing (which are assumed to be equal) and the z axis. For example, if the elevation is in units of meters and the grid values are separated by 10 meters, ‘zscale’ would be 10.

multicore Default ‘FALSE’. If ‘TRUE’, multiple cores will be used to compute the shadow matrix. By default, this uses all cores available, unless the user has set ‘options(“cores”)’ in which the multicore option will only use that many cores.

cache_mask Default ‘NULL’. A matrix of 1 and 0s, indicating which points on which the raytracer will operate.

shadow_cache Default ‘NULL’. The shadow matrix to be updated at the points defined by the argument ‘cache_mask’. If present, this will only compute the raytraced shadows for those points with value ‘1’ in the mask.

progbar Default ‘TRUE’ if interactive, ‘FALSE’ otherwise. If ‘FALSE’, turns off progress bar.

anglebreaks Default ‘NULL’. A vector of angle(s) in degrees (as measured from the horizon) specifying from where the light originates. Use this instead of ‘sunaltitude’ to create a softer shadow by specifying a wider light. E.g. ‘anglebreaks = seq(40,50,by=0.5)’ creates a light 10 degrees wide, as opposed to the default

Value

Matrix of light intensities at each point.
reduce_matrix_size

Examples

# First we ray trace the Monterey Bay dataset.
# The default angle is from 40-50 degrees azimuth, from the north east.

montereybay %>%
  ray_shade(zscale=50) %>%
  plot_map()

# Change the altitude of the sun to 25 degrees

montereybay %>%
  ray_shade(zscale=50, sunaltitude=25) %>%
  plot_map()

# Remove the lambertian shading to just calculate shadow intensity.

montereybay %>%
  ray_shade(zscale=50, sunaltitude=25, lambert=FALSE) %>%
  plot_map()

# Change the direction of the sun to the South East

montereybay %>%
  ray_shade(zscale=50, sunaltitude=25, sunangle=225) %>%
  plot_map()

---

reduce_matrix_size  Reduce Matrix Size

Description

Reduces the resolution of the matrix by specifying the desired output dimensions, or a scaling factor.

Usage

reduce_matrix_size(heightmap, scale = 0.5, width = NULL, height = NULL)

Arguments

- **heightmap**  The elevation matrix.
- **scale**  Default ‘0.5’. The amount to scale down the matrix. Scales down using bilinear interpolation.
- **width**  Default ‘NULL’. Alternative to ‘scale’ argument. The desired output width. If ‘width’ is less than 1, it will be interpreted as a scaling factor—e.g. 0.5 would halve the resolution for the width.
- **height**  Default ‘NULL’. Alternative to ‘scale’ argument. The desired output width. If ‘height’ is less than 1, it will be interpreted as a scaling factor—e.g. 0.5 would halve the resolution for the height.
render_camera

Examples

# Reduce the size of the monterey bay dataset by half
montbaysmall = reduce_matrix_size(montereybay, 0.5)
montbaysmall %>%
sphere_shade() %>%
plot_map()

# Reduce the size of the monterey bay dataset from 401x401 to 100x100
montbaysmall = reduce_matrix_size(montereybay, width = 100, height = 100)
montbaysmall %>%
sphere_shade() %>%
plot_map()

render_camera

Render Camera

Description

Changes the position and properties of the camera around the scene. If no values are entered, prints and returns the current values.

Usage

render_camera(theta = NULL, phi = NULL, zoom = NULL, fov = NULL)

Arguments

theta
phi
zoom
fov

Defaults to current value. Rotation angle.

Defaults to current value. Azimuth angle. Maximum ‘90’.

Defaults to current value. Positive value indicating camera magnification.

Defaults to current value. Field of view of the camera. Maximum ‘180’.

Examples

## Not run:
montereybay %>%
sphere_shade() %>%
plot_3d(montereybay,zscale = 50, water = TRUE, waterlinecolor="white")
render_snapshot()

## End(Not run)

# Shift the camera over and add a title
## Not run:
render_camera(theta = -45, phi = 45)
render_snapshot(title_text = “Monterey Bay, CA”,

## Not run:
render_camera(theta = 45, phi = -45)
render_snippet(title_text = “Monterey Bay, CA”,

## Not run:
render_camera(theta = 135, phi = 45)
render_snippet(title_text = “Monterey Bay, CA”,

# End(Not run)
# Shift to an overhead view (and change the text/title bar color)
## Not run:
render_camera(theta = 0, phi = 90, zoom = 0.9)
render_snapshot(title_text = "Monterey Bay, CA",
               title_color = "white",
               title_bar_color = "darkgreen")

## End(Not run)

# Shift to a front view and add a vignette effect
## Not run:
render_camera(theta = -90, phi = 30, zoom = 0.8)
render_snapshot(title_text = "Monterey Bay, CA",
               title_color = "white",
               title_bar_color = "blue",
               vignette = TRUE)

## End(Not run)

# Change the field of view (fov) and make the title bar opaque.
## Not run:
render_camera(theta = -90, phi = 30, zoom = 0.5, fov = 130)
render_snapshot(title_text = "Monterey Bay, CA",
               title_color = "black",
               title_bar_alpha = 1,
               title_bar_color = "lightblue",
               vignette = TRUE)

## End(Not run)

# Here we render a series of frames to later stitch together into a movie.
## Not run:
phivec = 20 + 70 * 1/(1 + exp(seq(-5, 10, length.out = 180)))
phivecfull = c(phivec, rev(phivec))

thetavec = 270 + 45 * sin(seq(0, 359, length.out = 360) * pi/180)
zoomvechalf = 0.5 + 0.5 * 1/(1 + exp(seq(-5, 10, length.out = 180)))
zoomvec = c(zoomvechalf, rev(zoomvechalf))

for(i in 1:360) {
  render_camera(theta = thetavec[i], phi = phivecfull[i], zoom = zoomvec[i])
  # uncomment the next line to save each frame to the working directory
  # render_snapshot(paste0("frame", i, ".png"))
}

# Run this command in the command line using ffmpeg to stitch together a video:
# ffmpeg -framerate 60 -i frame%d.png -vcodec libx264 raymovie.mp4

# And run this command to convert the video to post to the web:
# ffmpeg -i raymovie.mp4 -pix_fmt yuv420p -profile:v baseline -level 3 -vf scale=-2:-2 rayweb.mp4
render_compass

Render Compass Symbol

Description
Places a compass on the map to specify the North direction.

Usage
render_compass(
  angle = 0,
  position = "SE",
  x = NULL,
  y = NULL,
  z = NULL,
  compass_radius = NULL,
  scale_distance = 1,
  color_n = "darkred",
  color_arrow = "grey90",
  color_background = "grey60",
  color_bevel = "grey20",
  position_circular = FALSE,
  clear_compass = FALSE
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>angle</td>
<td>Default ‘0’.</td>
<td>The direction the arrow should be facing.</td>
</tr>
<tr>
<td>position</td>
<td>Default ‘SE’.</td>
<td>A string representing a cardinal direction. Ignored if ‘x’, ‘y’, and ‘z’ are manually specified.</td>
</tr>
<tr>
<td>x</td>
<td>Default ‘NULL’.</td>
<td>X position. If not entered, automatically calculated using ‘position’ argument.</td>
</tr>
</tbody>
</table>
render_compass

compass_radius Default 'NULL'. The radius of the compass. If not entered, automatically calculated. Increase or decrease the size of the compass.

scale_distance Default '1'. Multiplier that moves the compass away from the center of the map.

color_n Default 'darkred'. Color of the letter N.
color_arrow Default 'grey90'. Color of the arrow.
color_background Default 'grey20'. Color of the area right under the arrow.
color_bevel Default 'grey20'. Color of the bevel.
position_circular Default 'FALSE'. If 'TRUE', will place compass at a constant radius away from the map, as opposed to directly next to it. Overridden if user manually specifies position.
clear_compass Default 'FALSE'. Clears the compass symbol(s) on the map.

Value

Adds compass to map. No return value.

Examples

#Add a North arrow to the map, by default in the bottom right (SE)
## Not run:
montereybay %>%
  sphere_shade() %>%
  plot_3d(montereybay, theta=-45, water=TRUE)
render_compass()
render_snapshot()

#Remove the existing symbol with 'clear_compass = TRUE'
render_compass(clear_compass = TRUE)

#Point the N towards the light, at 315 degrees:
render_compass(angle = 315)
render_snapshot()
render_compass(clear_compass = TRUE)

#We can change the position by specifying a direction (here are three):
render_camera(theta=45, phi=45)
render_compass(position = "NW")
render_compass(position = "E")
render_compass(position = "S")
render_snapshot()
render_compass(clear_compass = TRUE)

#We can also change the distance away from the edge by setting the 'scale_distance' argument.
render_compass(position = "NW", scale_distance = 1.4)
render_compass(position = "E", scale_distance = 1.4)
render_compass(position = "S", scale_distance = 1.4)
#Zoom in slightly:
render_camera(theta=45, phi=45, zoom=0.7)
render_snapshot()
render_compass(clear_compass = TRUE)

# We can also specify the radius directly with `compass_radius`:
render_camera(theta=0, phi=45, zoom=1)
render_compass(position = "N", scale_distance = 1.5, compass_radius=200)
render_compass(position = "E", scale_distance = 1.4, compass_radius=50)
render_compass(position = "S", scale_distance = 1.3, compass_radius=25)
render_compass(position = "W", scale_distance = 1.2, compass_radius=10)
render_snapshot()
render_compass(clear_compass = TRUE)

# We can also adjust the position manually, by specifying all x, y and z arguments.
render_camera(theta=-45, phi=45, zoom=0.9)
render_compass(x = 150, y = 50, z = 150)
render_snapshot()

# Compass support is also included in render_highquality()
render_highquality()
render_compass(clear_compass = TRUE)

# We can change the colors in the compass, and also set it a constant distance away with
# `position_circular = TRUE`:
render_camera(theta=0, phi=45, zoom=0.75)
render_compass(position = "N", color_n = "#55967a", color_arrow = "#fff673",
              color_background = "#cfe0a9", color_bevel = "#8fb28a", position_circular = TRUE)
render_compass(position = "NE", color_n = "black", color_arrow = "grey90",
              color_background = "grey50", color_bevel = "grey20", position_circular = TRUE)
render_compass(position = "E", color_n = "red", color_arrow = "blue",
              color_background = "yellow", color_bevel = "purple", position_circular = TRUE)
render_compass(position = "SE", color_n = c(0.7,0.5,0.9), color_arrow = c(0.8,0.8,1),
              color_background = c(0.2,0.2,1), color_bevel = c(0.6,0.4,0.6),
              position_circular = TRUE)
render_compass(position = "S", color_n = "#ffe33b", color_arrow = "#6a463a",
              color_background = "#6ba698", color_bevel = "grey20", position_circular = TRUE)
render_compass(position = "SW", color_n = "#ffe33a", color_arrow = "#f1c3a9",
              color_background = "#bba9f8", color_bevel = "#66615e", position_circular = TRUE)
render_compass(position = "W", color_n = "#e9e671", color_arrow = "#cbb387",
              color_background = "#7c9695", color_bevel = "#cbb387", position_circular = TRUE)
render_compass(position = "NW", color_n = c(0.7,0,0), color_arrow = c(0.3,0,0),
              color_background = c(0.7,0.5,0.5), color_bevel = c(0.2,0,0), position_circular = TRUE)
render_snapshot(clear=TRUE)

## End(Not run)
render_depth

Description

Adding depth of field to the current RGL scene by simulating a synthetic aperture.

The size of the circle of confusion is determined by the following formula (z_depth is from the image’s depth map).

\[
\text{abs}(z_{\text{depth}}-\text{focus})\times f_{\text{ocal length}}^2/(f_{\text{stop}}\times z_{\text{depth}}(\text{focus}-f_{\text{ocal length}}))
\]

Usage

```r
render_depth(
  focus = 0.5,
  focallength = 100,
  fstop = 4,
  filename = NULL,
  preview_focus = FALSE,
  bokehshape = "circle",
  bokehintensity = 1,
  bokehlimit = 0.8,
  rotation = 0,
  gamma_correction = TRUE,
  aberration = 0,
  transparent_water = FALSE,
  heightmap = NULL,
  zscale = NULL,
  title_text = NULL,
  title_offset = c(20, 20),
  title_color = "black",
  title_size = 30,
  title_font = "sans",
  title_bar_color = NULL,
  title_bar_alpha = 0.5,
  image_overlay = NULL,
  vignette = FALSE,
  progbar = interactive(),
  instant_capture = interactive(),
  clear = FALSE,
  bring_to_front = FALSE,
  ...
)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>focus</td>
<td>Defaults ‘0.5’. Depth in which to blur. Minimum 0, maximum 1.</td>
</tr>
<tr>
<td>focallength</td>
<td>Default ‘1’. Focal length of the virtual camera.</td>
</tr>
<tr>
<td>fstop</td>
<td>Default ‘1’. F-stop of the virtual camera.</td>
</tr>
<tr>
<td>filename</td>
<td>The filename of the image to be saved. If this is not given, the image will be plotted instead.</td>
</tr>
</tbody>
</table>
preview_focus Default ‘FALSE’. If ‘TRUE’, a red line will be drawn across the image showing where the camera will be focused.
bokehintensity Default ‘3’. Intensity of the bokeh when the pixel intensity is greater than ‘bokehlimit’.
bokehlimit Default ‘0.8’. Limit after which the bokeh intensity is increased by ‘bokehintensity’.
rotation Default ‘0’. Number of degrees to rotate the hexagon bokeh shape.
gamma_correction Default ‘TRUE’. Controls gamma correction when adding colors. Default exponent of 2.2.
aberration Default ‘0’. Adds chromatic aberration to the image. Maximum of ‘1’.
transparent_water Default ‘FALSE’. If ‘TRUE’, depth is determined without water layer. User will have to re-render the water layer with ‘render_water()’ if they want to recreate the water layer.
heightmap Default ‘NULL’. The height matrix for the scene. Passing this will allow ‘render_depth()’ to automatically redraw the water layer if ‘transparent_water = TRUE’.
zs_scale Default ‘NULL’. The zs value for the heightmap. Passing this will allow ‘render_depth()’ to automatically redraw the water layer if ‘transparent_water = TRUE’.
title_text Default ‘NULL’. Text. Adds a title to the image, using magick::image_annotate.
title_offset Default ‘c(20,20)’. Distance from the top-left (default, ‘gravity’ direction in image_annotate) corner to offset the title.
title_color Default ‘black’. Font color.
title_size Default ‘30’. Font size in pixels.
title_font Default ‘sans’. String with font family such as "sans", "mono", "serif", "Times", "Helvetica", "Trebuchet", "Georgia", "Palatino" or "Comic Sans".
title_bar_color Default ‘NULL’. If a color, this will create a colored bar under the title.
title_bar_alpha Default ‘0.5’. Transparency of the title bar.
image_overlay Default ‘NULL’. Either a string indicating the location of a png image to overlay over the image (transparency included), or a 4-layer RGBA array. This image will be resized to the dimension of the image if it does not match exactly.
vignette Default ‘FALSE’. If ‘TRUE’ or numeric, a camera vignetting effect will be added to the image. ‘1’ is the darkest vignetting, while ‘0’ is no vignetting. If vignette is a length-2 vector, the second entry will control the blurriness of the vignette effect.
progbar Default ‘TRUE’ if in an interactive session. Displays a progress bar.
instant_capture
Default ‘TRUE’ if interactive, ‘FALSE’ otherwise. If ‘FALSE’, a slight delay is added before taking the snapshot. This can help stop prevent rendering issues when running scripts.

clear
Default ‘FALSE’. If ‘TRUE’, the current ‘rgl’ device will be cleared.
bring_to_front
Default ‘FALSE’. Whether to bring the window to the front when rendering the snapshot.
...
Additional parameters to pass to magick::image_annotate.

Value
4-layer RGBA array.

Examples

montereybay %>%
sphere_shade() %>%
plot_3d(montereybay,zscale=50, water=TRUE, waterlinecolor="white",
zoom=0.3, theta=-135, fov=70, phi=20)

# Preview where the focal plane lies
render_depth(focus=0.75, preview_focus=TRUE)

# Render the depth of field effect
render_depth(focus=0.75, focallength = 100)

# Add a chromatic aberration effect
render_depth(focus=0.75, focallength = 100, aberration = 0.3)

# Render the depth of field effect, ignoring water and re-drawing the waterlayer
render_depth(focus=0.9, preview_focus=TRUE,
heightmap = montereybay, zscale=50, transparent_water=TRUE)
render_depth(focus=0.9, heightmap = montereybay, zscale=50, transparent_water=TRUE)
rgl::rgl.close()

montereybay %>%
sphere_shade() %>%
plot_3d(montereybay,zscale=50, water=TRUE, waterlinecolor="white",
zoom=0.7, phi=30, fov=60, theta=-90)

render_camera(theta=45, zoom=0.15, phi=20)

# Change the bokeh shape and intensity
render_depth(focus=0.7, bokehshape = "circle", focallength=200, bokehintensity=30,
title_text = "Circular Bokeh", title_size = 20, title_color = "white",
title_bar_color = "white")
render_depth(focus=0.7, bokehshape = "hex", focallength=200, bokehintensity=30,
title_text = "Hexagonal Bokeh", title_size = 20, title_color = "white",
title_bar_color = "white")
render_highquality

Description

Renders a raytraced version of the displayed rgl scene, using the 'rayrender' package. User can specify the light direction, intensity, and color, as well as specify the material of the ground and add additional scene elements.

Usage

render_highquality(
    filename = NULL,
    light = TRUE,
    lightdirection = 315,
    lightaltitude = 45,
    lightsize = NULL,
    lightintensity = 500,
    lightcolor = "white",
    obj_material = rayrender::diffuse(),
    cache_filename = NULL,
    width = NULL,
    height = NULL,
    text_angle = NULL,
    text_size = 6,
    text_offset = c(0, 0, 0),
    line_radius = 0.5,
    scale_text_angle = NULL,
    scale_text_size = 6,
    scale_text_offset = c(0, 0, 0),
    title_text = NULL,
    title_offset = c(20, 20),
    title_color = "black",
    title_size = 30,
    title_font = "sans",
    title_bar_color = NULL,
    title_bar_alpha = 0.5,
    ground_material = rayrender::diffuse(),
    ground_size = 1e+05,
)
scene_elements = NULL,
camera_location = NULL,
camera_lookat = c(0, 0, 0),
camera_interpolate = 1,
clear = FALSE,
print_scene_info = FALSE,
...
)

Arguments

filename        Filename of saved image. If missing, will display to current device.
light           Default ‘TRUE’. Whether there should be a light in the scene. If not, the scene will be lit with a bluish sky.
lightdirection  Default ‘315’. Position of the light angle around the scene. If this is a vector longer than one, multiple lights will be generated (using values from ‘lightaltitude’, ‘lightintensity’, and ‘lightcolor’)
lightaltitude   Default ‘45°’. Angle above the horizon that the light is located. If this is a vector longer than one, multiple lights will be generated (using values from ‘lightdirection’, ‘lightintensity’, and ‘lightcolor’)
lightsize       Default ‘NULL’. Radius of the light(s). Automatically chosen, but can be set here by the user.
lightintensity  Default ‘500’. Intensity of the light.
lightcolor      Default ‘white’. The color of the light.
obj_material    Default ‘rayrender::diffuse()’. The material properties of the object file.
cache_filename  Name of temporary filename to store OBJ file, if the user does not want to rewrite the file each time.
width           Defaults to the width of the rgl window. Width of the rendering.
height          Defaults to the height of the rgl window. Height of the rendering.
text_angle      Default ‘NULL’, which forces the text always to face the camera. If a single angle (degrees), will specify the absolute angle all the labels are facing. If three angles, this will specify all three orientations (relative to the x,y, and z axes) of the text labels.
text_size       Default ‘6’. Height of the text.
text_offset     Default ‘c(0,0,0)’. Offset to be applied to all text labels.
line_radius     Default ‘0.5’. Radius of the label line segments.
scale_text_angle Default ‘NULL’. Same as ‘text_angle’, but for the scale bar.
scale_text_size Default ‘6’. Height of the scale bar text.
scale_text_offset Default ‘c(0,0,0)’. Offset to be applied to all scale bar text labels.
title_text      Default ‘NULL’. Text. Adds a title to the image, using magick::image_annotate.
title_offset Default `c(20,20)`. Distance from the top-left (default, ‘gravity’ direction in image_annotate) corner to offset the title.

title_color Default ‘black’. Font color.

title_size Default ‘30’. Font size in pixels.

title_font Default ‘sans’ String with font family such as “sans”, “mono”, “serif”, “times”, “helvetica”, “trebuchet”, “georgia”, “palatino” or “comic sans”.

title_bar_color Default ‘NULL’. If a color, this will create a colored bar under the title.

title_bar_alpha Default ‘0.5’. Transparency of the title bar.

ground_material Default ‘diffuse()’. Material defined by the rayrender material functions.

ground_size Default ‘100000’. The width of the plane representing the ground.

scene_elements Default ‘NULL’. Extra scene elements to add to the scene, created with rayrender.

camera_location Default ‘NULL’. Custom position of the camera. The ‘FOV’, ‘width’, and ‘height’ arguments will still be derived from the rgl window.

camera_lookat Default ‘NULL’. Custom point at which the camera is directed. The ‘FOV’, ‘width’, and ‘height’ arguments will still be derived from the rgl window.

camera_interpolate Default ‘c(0,0)’. Maximum ‘1’, minimum ‘0’. Sets the camera at a point between the ‘rgl’ view and the ‘camera_location’ and ‘camera_lookat’ vectors.

clear Default ‘FALSE’. If ‘TRUE’, the current ‘rgl’ device will be cleared.

print_scene_info Default ‘FALSE’. If ‘TRUE’, it will print the position and lookat point of the camera.

... Additional parameters to pass to rayrender::render_scene()

Examples

#Render the volcano dataset using pathtracing

```r
volcano %>%
sphere_shade() %>%
plot_3d(volcano,zscale = 2)
render_highquality()
```

#Change position of light

```r
render_highquality(lightdirection = 45)
```

#Change vertical position of light

```r
```
render_highquality(lightdirection = 45, lightaltitude=10)

# Change the ground material
render_highquality(lightdirection = 45, lightaltitude=60,
    ground_material = rayrender::diffuse(checkerperiod = 30, checkercolor="grey50"))

# Add three different color lights and a title
render_highquality(lightdirection = c(0,120,240), lightaltitude=45,
    lightcolor=c("red","green","blue"), title_text = "Red, Green, Blue",
    title_bar_color="white", title_bar_alpha=0.8)

# Change the camera:
render_camera(theta=-45,phi=60,fov=60,zoom=0.8)
render_highquality(lightdirection = c(0),
    title_bar_color="white", title_bar_alpha=0.8)

# Add a shiny metal sphere
render_camera(theta=-45,phi=60,fov=60,zoom=0.8)
render_highquality(lightdirection = c(0,120,240), lightaltitude=45,
    lightcolor=c("red","green","blue"),
    scene_elements = rayrender::sphere(z=-60,y=0,
        radius=20,material=rayrender::metal()))

# Add a red light to the volcano and change the ambient light to dusk
render_camera(theta=45,phi=45)
render_highquality(lightdirection = c(240), lightaltitude=30,
    lightcolor=c("#5555ff"),
    scene_elements = rayrender::sphere(z=0,y=15, x=-18, radius=5,
        material=rayrender::light(color="red",intensity=10)))

# Manually change the camera location and direction
render_camera(theta=45,phi=45,fov=90)
render_highquality(lightdirection = c(240), lightaltitude=30, lightcolor=c("#5555ff"),
    camera_location = c(50,10,10), camera_lookat = c(0,15,0),
    scene_elements = rayrender::sphere(z=0,y=15, x=-18, radius=5,
        material=rayrender::light(color="red",intensity=10)))

rgl::rgl.close()
Description

Adds a marker and label to the current 3D plot

Usage

render_label(
    heightmap,
    text,
    x,
    y,
    z,
    zscale = 1,
    relativez = TRUE,
    offset = 0,
    clear_previous = FALSE,
    textsizer = 1,
    dashed = FALSE,
    dashlength = "auto",
    linewidth = 3,
    antialias = FALSE,
    alpha = 1,
    textalpha = 1,
    freetype = TRUE,
    adjustvec = NULL,
    family = "sans",
    fonttype = "standard",
    linecolor = "black",
    textcolor = "black"
)

Arguments

heightmap  A two-dimensional matrix, where each entry in the matrix is the elevation at that point. All points are assumed to be evenly spaced.
text       The label text.
x           Either the ‘x’ coordinate in the matrix.
y           Either the ‘y’ coordinate in the matrix.
z           Elevation of the label, in units of the elevation matrix (scaled by zscale).
zscale     Default ‘1’. The ratio between the x and y spacing (which are assumed to be equal) and the z axis. For example, if the elevation levels are in units
relativez  Default ‘TRUE’. Whether ‘z’ should be measured in relation to the underlying elevation at that point in the heightmap, or set absolutely (‘FALSE’).
offset     Elevation above the surface (at the label point) to start drawing the line.
clear_previous  Default ‘FALSE’. If ‘TRUE’, it will clear all existing text and lines rendered with ‘render_label()’. If no other arguments are passed to ‘render_label()’, this will just remove all existing lines.
render_label

**textsize** Default ‘1’. A numeric character expansion value.
**dashed** Default ‘FALSE’. If ‘TRUE’, the label line is dashed.
**dashlength** Default ‘auto’. Length, in units of the elevation matrix (scaled by ‘zscale’) of the dashes if ‘dashed = TRUE’.
**linewidth** Default ‘3’. The line width.
**antialias** Default ‘FALSE’. If ‘TRUE’, the line with be have anti-aliasing applied. NOTE: anti-aliasing can cause some unpredictable behavior with transparent surfaces.
**alpha** Default ‘1’. Transparency of the label line.
**textalpha** Default ‘1’. Transparency of the label text.
**freetype** Default ‘TRUE’. Set to ‘FALSE’ if freetype is not installed (freetype enables anti-aliased fonts). NOTE: There are occasionally transparency issues when positioning FreeType fonts in front and behind a transparent surface.
**adjustvec** Default ‘c(0.5,-0.5)’. The horizontal and vertical offset for the text. If ‘freetype = FALSE’ and on macOS/Linux, this is adjusted to ‘c(0.33,-0.5)’ to keep the type centered.
**family** Default ‘“sans”’. Font family. Choices are ‘c(“serif”, “sans”, “mono”, “symbol”)’.
**fonttype** Default ‘“standard”’. The font type. Choices are ‘c(“standard”, “bold”, “italic”, “bolditalic”)’. NOTE: These require FreeType fonts, which may not be installed on your system. See the documentation for rgl::text3d() for more information.
**linecolor** Default ‘black’. Color of the line.
**textcolor** Default ‘black’. Color of the text.

**Examples**

```r
## Not run:
montereybay %>%
sphere_shade() %>%
plot_3d(montereybay,zscale=50,water=TRUE, watercolor="#233aa1")
render_snapshot()

## End(Not run)

#We want to add a label to Santa Cruz, so we use the x and y matrix coordinate (x=220 and y=330)
## Not run:
render_label(montereybay,x = 220, y = 330, z = 10000, zscale = 50, text = "Santa Cruz")
render_snapshot()

## End(Not run)

#We can also change the linetype to dashed by setting `dashed = TRUE` (additional options allow the user to control the dash length). You can clear the existing lines by setting `clear_previous = TRUE`.
## Not run:
render_label(montereybay, x = 290, y = 280, z = 10000, zs = 50, text = "Monterey",
          textcolor = "white", linecolor="darkred",dashed = TRUE, clear_previous = TRUE)
render_snapshot()
```
By default, \( z \) specifies the altitude above that point on the elevation matrix. We can also specify an absolute height by setting `relativez=FALSE`.

```r
render_label(montereybay, x=170, y=260, z=2000, text = "Monterey Canyon", relativez=FALSE)
render_snapshot()
```

We can also render labels in high quality with `render_highquality()`, specifying a custom line radius. By default, the labels point towards the camera, but you can fix their angle with argument `text_angle`.

```r
render_camera(theta=35, phi = 35, zoom = 0.80, fov=60)
render_label(montereybay, x = 290, y = 280, z = 10000, zscale = 50, text = "Monterey",
            textcolor = "white", linecolor="white",dashed = TRUE, clear_previous = TRUE)
render_label(montereybay,x=170,y=260, z=2000,zscale=50,textcolor = "white", linecolor="white",
            text = "Monterey Canyon",relativez=FALSE)
```

```r
render_highquality(samples=200,text_size = 18, line_radius = 2, text_offset = c(0,20,0),
            lightdirection=180, clamp_value=10)
```

```r
render_highquality(samples=200,text_size = 18, line_radius = 2, text_offset = c(0,20,0),
            lightdirection=180, text_angle=0, clamp_value=10)
```

We can remove all existing labels by calling `render_label(clear_previous = TRUE)`

```r
render_label(clear_previous = TRUE)
render_snapshot()
```

---

**render_movie**

**Render Movie**

Renders a movie using the `av` package. Moves the camera around a 3D visualization using either a standard orbit, or accepts vectors listing user-defined values for each camera parameter. If the latter, the values must be equal in length to `frames` (or of length `1`, in which the value will be fixed).

**Usage**

```r
render_movie(
    filename,
    type = "orbit",
)```
frames = 360,
fps = 30,
phi = 30,
theta = 0,
zoom = NULL,
fov = NULL,
title_text = NULL,
title_offset = c(20, 20),
title_color = "black",
title_size = 30,
title_font = "sans",
title_bar_color = NULL,
title_bar_alpha = 0.5,
image_overlay = NULL,
vignette = FALSE,
audio = NULL,
progbars = interactive(),
"

Arguments

filename Filename. If not appended with `.mp4`, it will be appended automatically.
type Default ‘orbit’, which orbits the 3D object at the user-set camera settings ‘phi’, ‘zoom’, and ‘fov’. Other options are ‘oscillate’ (sine wave around ‘theta’ value, covering 90 degrees), or ‘custom’ (which uses the values from the ‘theta’, ‘phi’, ‘zoom’, and ‘fov’ vectors passed in by the user).
frames Default ‘360’. Number of frames to render.
fps Default ‘30’. Frames per second. Recommend either 30 or 60 for web.
phi Defaults to current view. Azimuth values, in degrees.
theta Defaults to current view. Theta values, in degrees.
zoom Defaults to the current view. Zoom value, between ‘0’ and ‘1’.
fov Defaults to the current view. Field of view values, in degrees.
title_text Default ‘NULL’. Text. Adds a title to the movie, using `magick::image_annotate`.
title_offset Default ‘c(20,20)’. Distance from the top-left (default, ‘gravity’ direction in `image_annotate`) corner to offset the title.
title_color Default ‘black’. Font color.
title_size Default ‘30’. Font size in pixels.
title_bar_color Default ‘NULL’. If a color, this will create a colored bar under the title.
title_bar_alpha Default ‘0.5’. Transparency of the title bar.
image_overlay Default ‘NULL’. Either a string indicating the location of a png image to overlay over the whole movie (transparency included), or a 4-layer RGBA array. This image will be resized to the dimension of the movie if it does not match exactly.

vignette Default ‘FALSE’. If ‘TRUE’ or numeric, a camera vignetting effect will be added to the image. ‘1’ is the darkest vignetting, while ‘0’ is no vignetting. If vignette is a length-2 vector, the second entry will control the blurriness of the vignette effect.

audio Default ‘NULL’. Optional file with audio to add to the video.

progbar Default ‘TRUE’ if interactive, ‘FALSE’ otherwise. If ‘FALSE’, turns off progress bar. Will display a progress bar when adding an overlay or title.

... Additional parameters to pass to magick::image_annotate.

Examples

filename_movie = tempfile()

#By default, the function produces a 12 second orbit at 30 frames per second, at 30 degrees azimuth.

montereybay %>%
sphere_shade(texture="imhof1") %>%
plot_3d(montereybay, zscale=50, water = TRUE, watercolor="imhof1",
        waterlinecolor="white", waterlinealpha=0.5)
#Un-comment the following to run:
#render_movie(filename = filename_movie)

filename_movie = tempfile()

#You can change to an oscillating orbit. The magnification is increased and azimuth angle set to 30.
#A title has also been added using the title_text argument.

montereybay %>%
sphere_shade(texture="imhof1") %>%
plot_3d(montereybay, zscale=50, water = TRUE, watercolor="imhof1",
        waterlinecolor="white", waterlinealpha=0.5)
#Un-comment the following to run:
#render_movie(filename = filename_movie, type = "oscillate",
#frames = 60, phi = 30, zoom = 0.8, theta = -90,
#title_text = "Monterey Bay: Oscillating")

filename_movie = tempfile()

#Finally, you can pass your own set of values to the parameters as a vector with type = "custom".

phivechalf = 30 + 60 * 1/(1 + exp(seq(-7, 20, length.out = 180)/2))
phivecfull = c(phivechalf, rev(phivechalf))
thetavec = -90 + 45 * sin(seq(0,359,length.out = 360) * pi/180)
zoomvec = 0.45 + 0.2 * 1/(1 + exp(seq(-5, 20, length.out = 180)))
zoomvecfull = c(zoomvec, rev(zoomvec))

montereybay %>%
sphere_shade(texture="imhof1") %>
plot_3d(montereybay, zscale=50, water = TRUE, watercolor="imhof1",
       waterlinecolor="white", waterlinealpha=0.5)
#Un-comment the following to run
#render_movie(filename = filename_movie, type = "custom",
#   frames = 360, phi = phivecfull, zoom = zoomvecfull, theta = thetavec)
rgl::rgl.close()

---

render_scalebar  Render Scale Bar

Description

Places a compass on the map to specify the North direction.

Usage

render_scalebar(
  limits,
  position = "W",
  y = NULL,
  segments = 10,
  scale_length = 1,
  label_unit = "",
  offset = NULL,
  radius = NULL,
  color_first = "darkred",
  color_second = "grey80",
  color_text = "black",
  text_switch_side = FALSE,
  text_x_offset = 0,
  text_y_offset = 0,
  text_z_offset = 0,
  clear_scalebar = FALSE
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>limits</td>
<td>The distance represented by the scale bar. If a numeric vector greater than length 1, this will specify the breaks along the scale bar to place labels, with the maximum value in limits assumed to be the last label. Must be non-negative.</td>
</tr>
<tr>
<td>position</td>
<td>Default 'W'. A string representing a direction. Can be 'N', 'E', 'S', and 'W'.</td>
</tr>
<tr>
<td>y</td>
<td>Default 'NULL'. The height of the scale bar, automatically calculated if 'NULL'.</td>
</tr>
<tr>
<td>segments</td>
<td>Default '10'. Number of colored segments in the scalebar.</td>
</tr>
</tbody>
</table>
scale_length: Default ‘1’. Length of the scale bar, relative to the side of the map specified in ‘position’. If a length-2 vector, the first number specifies the start and stop points along the side.

label_unit: Default ‘NULL’. The distance unit for the label.

offset: Default ‘NULL’. The distance away from the edge to place the scale bar. If ‘NULL’, automatically calculated.

radius: Default ‘NULL’. The radius of the cylinder representing the scale bar. If ‘NULL’, automatically calculated.

color_first: Default ‘darkred’. Primary color in the scale bar.

color_second: Default ‘grey90’. Secondary color in the scale bar.

color_text: Default ‘black’. Color of the text.

text_switch_side: Default ‘FALSE’. Switches the order of the text.

text_x_offset: Default ‘0’. Distance offset for text in the x direction.

text_y_offset: Default ‘0’. Distance offset for text in the y direction.

text_z_offset: Default ‘0’. Distance offset for text in the z direction.

clear_scalebar: Default ‘FALSE’. Clears the scale bar(s) on the map.

Value
Displays snapshot of current rgl plot (or saves to disk).

Examples

```r
# Add a scale bar to the montereybay dataset, here representing about 80km
## Not run:
montereybay %>%
sphere_shade() %>%
plot_3d(montereybay, theta=45, water=TRUE)
render_scalebar(limits=c(0, 80), label_unit = "km")
render_snapshot()

# This function works with `render_highquality()`

render_highquality(lightdirection=250, lightaltitude=40, scale_text_size=24, clamp_value=10)
render_scalebar(clear_scalebar = TRUE)

# We can change the position by specifying a cardinal direction to 'position', and the
# color by setting 'color_first' and 'color_second'

render_scalebar(limits=c(0,80), label_unit = "km", position = "N",
               color_first = "darkgreen", color_second = "lightgreen")
render_snapshot()
render_scalebar(clear_scalebar = TRUE)

# And switch the orientation by setting 'text_switch_side = TRUE'
render_scalebar(limits=c(0,80), label_unit = "km", position = "N", text_switch_side = TRUE,
               color_first = "darkgreen", color_second = "lightgreen")
```
render_snapshot()
render_scalebar(clear_scalebar = TRUE)

# We can add additional breaks by specifying additional distances in `limits`

render_scalebar(limits=c(0,40,80), label_unit = "km")
render_snapshot()
render_scalebar(clear_scalebar = TRUE)

# We can also manually specify the height by setting the `y` argument:

render_scalebar(limits=c(0,40,80), y=-70, label_unit = "km")
render_snapshot()
render_scalebar(clear_scalebar = TRUE)

# Here we change the total size by specifying a start and end point along the side,
# and set the number of colored `segments`:

render_scalebar(limits=c(0,20, 40), segments = 4, scale_length = c(0.5,1), label_unit = "km")
render_scalebar(limits=c(0,20, 40), segments = 4, position = "N", text_switch_side = TRUE,
               scale_length = c(0.25,0.75), label_unit = "km")
render_snapshot()
render_scalebar(clear_scalebar = TRUE)

# Change the radius of the scale bar with `radius`. Here, the autopositioning doesn’t work well with
# the labels, so we provide additional offsets with `text_y_offset` and `text_x_offset` to fix it.

render_scalebar(limits=c(0,20, 40), segments = 4, scale_length = c(0.5,1),
               label_unit = "km", radius=10,text_y_offset=-20,text_x_offset=20)
render_snapshot()
render_scalebar(clear_scalebar = TRUE)

## End(Not run)

---

render_snapshot  Render Snapshot of 3D Visualization

**Description**

Either captures the current rgl view and displays, or saves the current view to disk.

**Usage**

```r
render_snapshot(
  filename,
  clear = FALSE,
  title_text = NULL,
  title_offset = c(20, 20),
  title_color = "black",
  title_size = 30,
)```
title_font = "sans",
title_bar_color = NULL,
title_bar_alpha = 0.5,
image_overlay = NULL,
vignette = FALSE,
instant_capture = interactive(),
bring_to_front = FALSE,
...)

Arguments

filename Filename of snapshot. If missing, will display to current device.
clear Default ‘FALSE’. If ‘TRUE’, the current ‘rgl’ device will be cleared.
title_text Default ‘NULL’. Text. Adds a title to the image, using magick::image_annotate.
title_offset Default ‘c(20,20)’. Distance from the top-left (default, ‘gravity’ direction in image_annotate) corner to offset the title.
title_color Default ‘black’. Font color.
title_size Default ‘30’. Font size in pixels.
title_font Default ‘sans’. String with font family such as "sans", "mono", "serif", "Times", "Helvetica", "Trebuchet", "Georgia", "Palatino” or "Comic Sans".
title_bar_color Default ‘NULL’. If a color, this will create a colored bar under the title.
title_bar_alpha Default ‘0.5’. Transparency of the title bar.
image_overlay Default ‘NULL’. Either a string indicating the location of a png image to overlay over the image (transparency included), or a 4-layer RGBA array. This image will be resized to the dimension of the image if it does not match exactly.
vignette Default ‘FALSE’. If ‘TRUE’ or numeric, a camera vignetting effect will be added to the image. ‘1’ is the darkest vignetting, while ‘0’ is no vignetting. If vignette is a length-2 vector, the second entry will control the blurriness of the vignette effect.
instant_capture Default ‘TRUE’ if interactive, ‘FALSE’ otherwise. If ‘FALSE’, a slight delay is added before taking the snapshot. This can help stop prevent rendering issues when running scripts.
bring_to_front Default ‘FALSE’. Whether to bring the window to the front when taking the snapshot.
...
... Additional parameters to pass to magick::image_annotate.

Value

Displays snapshot of current rgl plot (or saves to disk).
render_water

## Description

Adds water layer to the scene, removing the previous water layer if desired.

## Usage

```r
render_water(
  heightmap,
  waterdepth = 0,
  watercolor = "lightblue",
  zscale = 1,
  wateralpha = 0.5,
  waterlinecolor = NULL,
)```
waterlinealpha = 1,
linewidth = 2,
remove_water = TRUE
)

Arguments

heightmap A two-dimensional matrix, where each entry in the matrix is the elevation at that point. All points are assumed to be evenly spaced.
waterdepth Default ‘0’.
watercolor Default ‘lightblue’.
zs
scale Default ‘1’. The ratio between the x and y spacing (which are assumed to be equal) and the z axis. For example, if the elevation levels are in units of 1 meter and the grid values are separated by 10 meters, ‘zs
cale’ would be 10.
wateralpha Default ‘0.5’. Water transparency.
waterlinecolor Default ‘NULL’. Color of the lines around the edges of the water layer.
waterlinealpha Default ‘1’. Water line transparency.
linewidth Default ‘2’. Width of the edge lines in the scene.
remove_water Default ‘TRUE’. If ‘TRUE’, will remove existing water layer and replace it with new layer.

Examples

## Not run:
montereybay %>%
sphere_shade() %>%
plot_3d(montereybay,zscale=50)
render_snapshot()

## End(Not run)

#We want to add a layer of water after the initial render.
## Not run:
render_water(montereybay,zscale=50)
render_snapshot()

## End(Not run)

#Call it again to change the water depth
## Not run:
render_water(montereybay,zscale=50,waterdepth=-1000)
render_snapshot()

## End(Not run)

#Add waterlines
## Not run:
render_camera(theta=-45)
render_water(montereybay,zscale=50,waterlinecolor="white")
Description

Writes a stereolithography (STL) file that can be used in 3D printing.

Usage

save_3dprint(
  filename,
  maxwidth = 125,
  unit = "mm",
  rotate = TRUE,
  remove_extras = TRUE,
  clear = FALSE
)

Arguments

filename String with the filename. If ‘.stl’ is not at the end of the string, it will be appended automatically.

maxwidth Default ‘125’. Desired maximum width of the 3D print in millimeters. Uses the units set in ‘unit’ argument. Can also pass in a string, "125mm" or "5in".

unit Default ‘mm’. Units of the ‘maxwidth’ argument. Can also be set to inches with ‘in’.

rotate Default ‘TRUE’. If ‘FALSE’, the map will be printing on its side. This may improve resolution for some 3D printing types.

remove_extras Default ‘TRUE’. Removes non-topographic features from base: lines, water, labels, and the shadow.

clear Default ‘FALSE’. If ‘TRUE’, the current ‘rgl’ device will be cleared.

Value

Writes an STL file to ‘filename’. Regardless of the unit displayed, the output STL is in millimeters.
Examples

```r
filename_stl = tempfile()

#Save the STL file into `filename_stl`
volcano %>%
  sphere_shade() %>%
  plot_3d(volcano,zscale=3)
render_snapshot()
save_3dprint(filename_stl, clear=TRUE)

#Save the STL file into `filename_stl`, setting maximum width to 100 mm
volcano %>%
  sphere_shade() %>%
  plot_3d(volcano,zscale=3)
render_snapshot()
save_3dprint(filename_stl, maxwidth = 100, clear=TRUE)

## Save the STL file into `filename_stl`, setting maximum width to 4 inches
volcano %>%
  sphere_shade() %>%
  plot_3d(volcano,zscale=3)
render_snapshot()
save_3dprint(filename_stl, maxwidth = 4, unit = "in", clear=TRUE)

## Save the STL file into `filename_stl`, setting maximum width (character) to 120mm
volcano %>%
  sphere_shade() %>%
  plot_3d(volcano,zscale=3)
render_snapshot()
save_3dprint(filename_stl, maxwidth = "120mm", clear=TRUE)
```

### save_obj

**Save OBJ**

**Description**

Writes the textured 3D rayshader visualization to an OBJ file.

**Usage**

```r
save_obj(filename, save_texture = TRUE, water_index_refraction = 1)
```
save_png

Arguments

filename String with the filename. If `.obj` is not at the end of the string, it will be appended automatically.

save_texture Default ‘TRUE’. If the texture should be saved along with the geometry.

water_index_refraction Default ‘1’. The index of refraction for the rendered water.

Examples

filename_obj = tempfile(fileext = ".obj")

#Save model of volcano

volcano %>%
sphere_shade() %>%
plot_3d(volcano, zscale = 2)

save_obj(filename_obj)

#Save model of volcano without texture

save_obj(filename_obj, save_texture = FALSE)
rl::rl.close()

#Make water have realistic index of refraction

montereybay %>%
sphere_shade() %>%
plot_3d(montereybay, zscale = 50)

save_obj(filename_obj, water_index_refraction = 1.5)
rl::rl.close()

Description

Writes the hillshaded map to file.

Usage

save_png(hillshade, filename, rotate = 0, dpi = NULL)
**sphere_shade**

**Calculate Surface Color Map**

**Description**

Calculates a color for each point on the surface using the surface normals and hemispherical UV mapping. This uses either a texture map provided by the user (as an RGB array), or a built-in color texture.

**Usage**

```r
sphere_shade(
  heightmap,
  sunangle = 315,
  texture = "imhof1",
  normalvectors = NULL,
  colorintensity = 1,
  zscale = 1,
  progbar = interactive()
)
```
sphere_shade

Arguments

heightmap A two-dimensional matrix, where each entry in the matrix is the elevation at that point. All points are assumed to be evenly spaced.
sunangle Default ‘315° (NW). The direction of the main highlight color (derived from the built-in palettes or the ‘create_texture’ function).
texture Default ‘imhof1’. Either a square matrix indicating the spherical texture mapping, or a string indicating one of the built-in palettes (‘imhof1’, ‘imhof2’, ‘imhof3’, ‘imhof4’, ‘desert’, ‘bw’, and ‘unicorn’).
normalvectors Default ‘NULL’. Cache of the normal vectors (from ‘calculate_normal’ function). Supply this to speed up texture mapping.
colorintensity Default ‘1’. The intensity of the color mapping. Higher values will increase the intensity of the color mapping.
zscale Default ‘1/colorintensity‘. The ratio between the x and y spacing (which are assumed to be equal) and the z axis. Ignored unless ‘colorintensity’ missing.
probar Default ‘TRUE’ if interactive, ‘FALSE’ otherwise. If ‘FALSE’, turns off progress bar.

Value

RGB array of hillshaded texture mappings.

Examples

# Basic example:
montereybay %>%
sphere_shade() %>%
plot_map()

# Decrease the color intensity:
montereybay %>%
sphere_shade(colorintensity=0.1) %>%
plot_map()

# Change to a built-in color texture:
montereybay %>%
sphere_shade(texture="desert") %>%
plot_map()

# Change the highlight angle:
montereybay %>%
sphere_shade(texture="desert", sunangle = 45) %>%
plot_map()

# Create our own texture using the `create_texture` function:
montereybay %>%
sphere_shade(texture=create_texture("springgreen","darkgreen",
                                    "turquoise","steelblue3","white")) %>%
plot_map()
re-export magrittr pipe operator

Description

re-export magrittr pipe operator
Index

*Topic datasets

  montereybay, 12

%>%, 52

detect_water, 9
height_shade, 10
lamb_shade, 11
montereybay, 12
plot_3d, 13
plot_gg, 16
plot_map, 20
raster_to_matrix, 21
ray_shade, 21
reduce_matrix_size, 23
render_camera, 24
render_compass, 26
render_depth, 28
render_highquality, 32
render_label, 35
render_movie, 38
render_scalebar, 41
render_snapshot, 43
render_water, 45
save_3dprint, 47
save_obj, 48
save_png, 49
sphere_shade, 50