Package 'rayshader'

July 11, 2019

Type Package

Title Create and Visualize Hillshaded Maps from Elevation Matrices

Version 0.11.5

Author Tyler Morgan-Wall

Maintainer Tyler Morgan-Wall <tylermw@gmail.com>

Description Uses a combination of raytracing, spherical texture mapping, lambertian reflectance, and ambient occlusion 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, and the ability to export 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, imager, grDevices, ggplot2, grid, utils

Suggests reshape2, viridis, av

LinkingTo Rcpp, progress, RcppArmadillo

RoxygenNote 6.1.1

URL https://github.com/tylermorganwall/rayshader

BugReports https://github.com/tylermorganwall/rayshader/issues

NeedsCompilation yes

Repository CRAN

Date/Publication 2019-07-11 13:43:03 UTC

R topics documented:

`add_overlay` *Add Overlay*

Description

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

Usage

```
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 interpretted as completely transparent.
`alphalayer`	Default ‘1‘. Defines minimum tranparaency 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 tranparencies 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

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

fliplr = function(x) {
 x[,ncol(x):1]
}
## Not run:
tempfilename = tempfile()
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

```
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 incidates 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

```
#Raytrace the `volcano` elevation map and add that shadow to the output of sphere_shade()
shadowmap = ray_shade(volcano)

volcano %>%
 sphere_shade() %>%
 add_shadow(shadowmap) %>%
 plot_map()

#Increase the intensity of the shadow:
volcano %>%
 sphere_shade() %>%
 add_shadow(shadowmap,0.3) %>%
 plot_map()
```

add_water *Add Water*

Description

Adds a layer of water to a map.

Usage

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

Arguments

hillshade	A three-dimensional RGB array.
watermap	Matrix indicating whether water was detected at that point. 1 indicates water, 0 indicates no water.
color	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').

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()
```

ambient_shade *Calculate Ambient Occlusion Map*

Description

Calculates Ambient Occlusion Shadow Map

Usage

```
ambient_shade(heightmap, anglebreaks = seq(1, 46, 15), sunbreaks = 12,
  maxsearch = 20, 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	The angle(s), in degrees, as measured from the horizon from which the light originates.
sunbreaks	Default 12. Number of rays to be sent out in a circle, evenly spaced, around the point being tested.
maxsearch	Default '20'. The maximum horizontal distance that the system should propogate 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.
zscale	Default 1. The ratio between the x and y spacing (which are assumed to be equal) and the z axis.
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'.
progbar	Default 'TRUE' if interactive, 'FALSE' otherwise. If 'FALSE', turns off progress bar.
...	Additional arguments to pass to the 'makeCluster' function when 'multicore=TRUE'.

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)
```

calculate_normal *Calculate Normal*

Description

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

Usage

```
calculate_normal(heightmap, zscale = 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.
zscale	Default 1.
progbar	Default 'FALSE'. If 'TRUE', turns on progress bar.

Value

Matrix of light intensities at each point.

Examples

```
#Here we produce a light intensity map of the `volcano` elevation map.

#Cache the normal vectors of the volcano dataset
volcanocache = calculate_normal(volcano)

#Use the cached vectors to speed up calculation of `sphere_shade()` on a map.
sphere_shade(volcano,normalvectors = volcanocache) %>%
 plot_map()
```

create_texture *Create Texture*

Description

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

Usage

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

Arguments

lightcolor	The main highlight color. Corresponds to the top center of the texture map.
shadowcolor	The main shadow color. Corresponds to the bottom center of the texture map. This color represents slopes directed directly opposite to the main highlight color.
leftcolor	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.
rightcolor	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.
centercolor	The center color. Corresponds to the center of the texture map. This color represents flat areas.
cornercolors	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.

Examples

```
#Here is the `imhof1` 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

```
detect_water(heightmap, zscale = 1, cutoff = 0.999,
  min_area = length(heightmap)/400, 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.
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.

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()
```

---

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 ve

Usage

```
lamb_shade(heightmap, rayangle = 45, sunangle = 315, zscale = 1,
  zero_negative = 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.
rayangle	Default '45'. The azimuth angle as measured from the horizon from which the light originates.

sunangle	Default '315' (NW). The angle around the matrix from which the light originates.
zscale	Default '1'. The ratio between the x and y spacing (which are assumed to be equal) and the z axis.
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,
   rayangle = 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

```
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", 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 tranparency.
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.
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.
...	Additional arguments to pass to the 'rgl::par3d' function.

Examples

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

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


#With a water layer

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"

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)
```

```
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)


#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.

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


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

---

plot_gg *Transform ggplot2 objects into 3D*

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

```
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,
  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.

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)) +
 scale_fill_gradientn("Z",colours = terrain.colors(10)) +
 coord_fixed()
ggvolcano


plot_gg(ggvolcano, multicore = TRUE, raytrace = TRUE, width = 7, height = 4,
       scale = 300, windowsize = c(1400, 866), zoom = 0.6, phi = 30, theta = 30)
render_snapshot(clear = TRUE)
```

```
#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`.
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.

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



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


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


plot_gg(mtplot_density, width = 4,zoom = 0.60, theta = -45, phi = 30,
       windowsize = c(1400,866))
render_snapshot(clear = TRUE)


#This also works facetted.
mtplot_density_facet = mtplot_density + facet_wrap(~cyl)

#Preview this plot in 2D:

plot_gg(mtplot_density_facet, preview = TRUE)



plot_gg(mtplot_density_facet, windowsize=c(1400,866),
       zoom = 0.55, theta = -10, phi = 25)
render_snapshot(clear = TRUE)


#That is a little cramped. Specifying a larger width will improve the readability of this plot.

plot_gg(mtplot_density_facet, width = 6, preview = TRUE)


#That's better. Let's plot it in 3D, and increase the scale.
```

```
plot_gg(mtplot_density_facet, width = 6, windowsize=c(1400,866),
       zoom = 0.55, theta = -10, phi = 25, scale=300)
render_snapshot(clear = TRUE)
```

---

plot_map *Plot Map*

Description

Displays the map in the current device.

Usage

```
plot_map(hillshade, rotate = 0, ...)
```

Arguments

hillshade	Hillshade to be plotted.
rotate	Default 0. Rotates the output. Possible values: 0, 90, 180, 270.
...	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))
```

---

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

```
ray_shade(heightmap, anglebreaks = seq(40, 50, 1), sunangle = 315,
  maxsearch = 100, lambert = TRUE, zscale = 1, multicore = FALSE,
  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 'seq(40,50,1)'. The azimuth angle(s), in degrees, as measured from the horizon from which the light originates.
sunangle	Default '315' (NW). The angle, in degrees, around the matrix from which the light originates. Zero degrees is North, increasing clockwise.
maxsearch	Default '100'. The maximum distance that the system should propogate rays to check. For longer
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 levels are in units of 1 meter 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.
...	Additional arguments to pass to the 'makeCluster' function when 'multicore=TRUE'.

Value

Matrix of light intensities at each point.

Examples

```
#Here we produce an shadow map of the `volcano` elevation map with the light from the NE.
#The default angle is from 40-50 degrees azimuth, from the north east.
volcanoshadow = ray_shade(volcano)

#Turn off Lambertian shading to get a shadow map solely based on the raytraced shadows.
volcanoshadow = ray_shade(heightmap = volcano,
   anglebreaks = seq(30,40,10),
   sunangle = 45,
   maxsearch = 100,
   lambert = FALSE)

plot_map(volcanoshadow)
```

render_camera *Render Camera*

Description

Changes the position and properties of the camera around the scene. Wrapper around rgl.viewpoint.

Usage

```
render_camera(theta = 45, phi = 45, zoom = NULL, fov = NULL)
```

Arguments

theta	Default '45'. Rotation angle.
phi	Default '45'. Azimuth angle. Maximum '90'.
zoom	Defaults to current value. Positive value indicating camera magnification.
fov	Defaults to current value. Field of view of the camera. Maximum '180'.

Examples

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

## End(Not run)

#Shift the camera over
## Not run:
render_camera(theta = -45, phi = 45)
render_snapshot()

## End(Not run)

#Shift to an overhead view
## Not run:
render_camera(theta = 0, phi = 90,zoom = 0.7)
render_snapshot()

## End(Not run)

#Shift to an front view
## Not run:
render_camera(theta = -90, phi = 30,zoom = 0.5)
render_snapshot()

## End(Not run)
```

```
#Change the FOV
## Not run:
render_camera(theta = -90, phi = 30,zoom = 0.5,fov = 130)
render_snapshot()
rgl::rgl.close()

## End(Not run)

#Here we render a series of frames to later stitch together into a movie.
## Not run:
montereybay %>%
 sphere_shade() %>%
 plot_3d(montereybay,zscale = 50)

phivec = 20 + 70 * 1/(1 + exp(seq(-5, 10, length.out = 180)))
phivecfull = c(phivec, rev(phivec))
thetavec = 270 + 90 * sin(seq(0,359,length.out = 360) * pi/180)
zoomvec = 0.5 + 0.5 * 1/(1 + exp(seq(-5, 10, length.out = 180)))
zoomvecfull = c(zoomvec, rev(zoomvec))

for(i in 1:360) {
 render_camera(theta = thetavec[i],phi = phivecfull[i],zoom = zoomvecfull[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

## End(Not run)
```

`render_depth` *Render Depth of Field*

Description

Adds 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).

```
abs(z_depth-focus)*focal_length^2/(f_stop*z_depth*(focus - focal_length))
```

Usage

```
render_depth(focus = 0.5, focallength = 100, fstop = 4,
  filename = NULL, bokehshape = "circle", bokehintensity = 1,
  bokehlimit = 0.8, rotation = 0, gamma_correction = TRUE,
  transparent_water = FALSE, progbar = interactive(), clear = FALSE)
```

Arguments

focus	Defaults '0.5'. Depth in which to blur. Minimum 0, maximum 1.
focallength	Default '1'. Focal length of the virtual camera.
fstop	Default '1'. F-stop of the virtual camera.
filename	The filename of the image to be saved. If this is not given, the image will be plotted instead.
bokehshape	Default 'circle'. Also built-in: 'hex'. The shape of the bokeh.
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.
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.
progbar	Default 'TRUE' if in an interactive session. Displays a progress bar.
clear	Default 'FALSE'. If 'TRUE', the current 'rgl' device will be cleared.

Value

4-layer RGBA array.

Examples

```
montereybay %>%
 sphere_shade() %>%
 plot_3d(montereybay,zscale=50,zoom=0.6,theta=-90)

render_depth(focallength = 30)
render_depth(focallength = 30,fstop=2)
render_depth(focallength = 30,fstop=2, clear = TRUE)
```

render_label *Render Label*

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, textsize = 1, dashed = FALSE, dashlength = "auto",
  linewidth = 3, antialias = FALSE, alpha = 1, textalpha = 1,
  freetype = TRUE, adjustvec = NULL, family = "sans",
  fonttype = "standard", color = "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	Text ‘x‘ coordinate in the matrix.
y	Text ‘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.
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.
color	Default ‘black‘. Color of the line.
textcolor	Default ‘black‘. Color of the text.

Examples

```
## Not run:
montereybay %>%
 sphere_shade() %>%
 plot_3d(montereybay,zscale=50,water=TRUE)
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)
## Not run:
render_label(montereybay,x=300,y=120, z=10000,zscale=50,text = "Monterey",dashed=TRUE)
render_snapshot()

## End(Not run)

#By default, z specifies the altitude above that point on the elevation matrix. We can also specify
#an absolute height by setting `relativez=FALSE`.
## Not run:
render_label(montereybay,x=50,y=130, z=2000,zscale=50,text = "Monterey Canyon",relativez=FALSE)
render_snapshot(clear = TRUE)

## End(Not run)
```

---

render_movie *Render Movie*

Description

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

```
render_movie(filename, type = "orbit", frames = 360, fps = 30,
  phi = 30, theta = 0, zoom = NULL, fov = NULL, ...)
```

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. Recommmend either 30 or 60 for web.
phi	Default '30'. Azimuth values, in degrees.
theta	Default '0'. 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.
...	Additional parameters to pass to av::av_capture_graphics

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.

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)

filename_movie = tempfile()

#Finally, you can pass your own set of values to the
#camera 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 + 60 * 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)
```

`render_snapshot` *Render Snapshot of 3D Visualization*

Description

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

Usage

```
render_snapshot(filename, clear = 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.

Value

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

Examples

```
## Not run:
montereybay %>%
 sphere_shade() %>%
 plot_3d(montereybay,zscale=50,zoom=0.6,theta=-90)

## End(Not run)

## Not run:
render_snapshot(clear = TRUE)

## End(Not run)
```

render_water *Render Water Layer*

Description

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

Usage

```
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‘.
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.
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 tranparency.
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(clear = TRUE)

## End(Not run)
```

---

save_3dprint *Save 3D Print*

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

```
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_png *Save PNG*

Description

Writes the hillshaded map to file.

Usage

```
save_png(hillshade, filename, rotate = 0, dpi = NULL)
```

Arguments

hillshade	Array (or matrix) of hillshade to be written.
filename	String with the filename. If '.png' is not at the end of the string, it will be appended automatically.
rotate	Default 0. Rotates the output. Possible values: 0, 90, 180, 270.
dpi	Default 'NULL'. Optional DPI (dots per inch) specifying the resolution of the image.

Examples

```
filename_map = tempfile()

#Save the map into `filename_map`
montereybay %>%
 sphere_shade() %>%
 save_png(filename_map)

#Rotate the map 180 degrees:

montereybay %>%
 sphere_shade() %>%
 save_png(filename_map,rotate=180)
```

---

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

```
sphere_shade(heightmap, sunangle = 315, texture = "imhof1",
  normalvectors = NULL, zscale = 1, 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.
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.
zscale	Default '1'. The ratio between the x and y spacing (which are assumed to be equal) and the z axis.
progbar	Default 'TRUE' if interactive, 'FALSE' otherwise. If 'FALSE', turns off progress bar.

Value

RGB array of hillshaded texture mappings.

Examples

```
montereybay %>%
 sphere_shade() %>%
 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