Package 'rglplus'

January 21, 2023

Type Package

Title Extension of the 'rgl' 3D Visualization Package

Version 1.1

Author Danail Obreschkow

Maintainer Danail Obreschkow <danail.obreschkow@gmail.com>

Description Provides 3D plotting routines that facilitate the use of the 'rgl' package and extend its functionality. For example, the routines allow the user to directly control the camera position & orientation, as well as to generate 3D movies with a moving observer.

Imports rgl, stats

License GPL-3

Encoding UTF-8

RoxygenNote 7.2.3

NeedsCompilation no

Repository CRAN

Date/Publication 2023-01-21 14:40:03 UTC

R topics documented:

rgl.ball *Plot high-resolution sphere or globe*

Description

Draws a sphere in custom resolution with custom surface image.

Usage

```
rgl.ball(
  x = 0,
  y = 0,
  z = 0,
  radius = 1,
  depth = 5,
  png = NULL,
  rotation = NULL,
  normals = "standard",
  ...
)
```

Arguments

x	x-coordinate of the center of the sphere
y	y-coordinate of the center of the sphere
z	z-coordinate of the center of the sphere
radius	radius of the sphere
depth	integer (1...8) specifying the number of rectangles (=6*4^depth)
png	optional character string specifying the file name of a png-image to be rendered on the sphere. This file must contain the map to be displayed in an equirectangular projection (also known as equidistant cylindrical projection).
rotation	optional 3-by-3 or or 4-by-4 rotation matrix applied to the whole globe; only used of png is specified.
normals	character string specifying the way the normal vectors of the surface are internally passed to surface3d. This argument is available because surface3d (or rather the underlying routine rgl.surface) sometimes handles the sign of normal vectors incorrectly, causing light sources to appear in the wrong direction if a rotation matrix is provided. The argument can take three values: "none" does not pass any normal vectors to rgl.surface, hence avoiding any issues with the direction of light sources, but this can cause glitches at the 180-degree meridian (choose a high depth around 7 in this case); "standard" passes correct normal vectors to rgl.surface, which can cause wrong lighting for certain rotation matrices; "improved" is similar to "standard", but corrects the lighting errors in most cases.
...	additional parameter to refine the material properties (see rgl.material).

Value

None

Author(s)

Danail Obreschkow (thanks to input from Aaron Robotham's `sphereplot` package)

Examples

```
# Show Earth with core
rgl::open3d()
rgl.ball(0, 0, 0, 1, png=system.file('earth.png', package='rglplus'), emission='grey', alpha=0.6)
rgl.ball(0, 0, 0, 0.6, col='red')
```

`rgl.camera` *Place observer*

Description

Set the position, orientation and field-of-view of the observer

Usage

```
rgl.camera(position = NULL, direction = NULL, up = NULL, fov = 0)
```

Arguments

`position`	either a 3-vector, a single number or NULL. A vector directly specifies the location of camera. A single number specifies the distances of the camera along the z-axis, relative to the center of the scene (= center of the bounding box returned by `par3d`). If no positions is given, then the camera is placed automatically.
`direction`	optional 3-vector specifying the direction in which the observer is looking, that is the optical axis of the virtual camera. The norm of the vector is irrelevant, but has to be non-zero. If not given, the camera is pointed at the center of the scene.
`up`	optional single number or 3-vector, specifying the rotation of the camera around the optical axis (as defined with the argument direction). If a single number is provided, it is normally interpreted as the angle in degrees between the up-direction of the 2d camera image and the projected z-axis of the 3d scene. To avoid the singularity that occurs if the optical axis lies very close to the z-axis, "up" is, in this case, interpreted as the angle between the up-direction and the y-axis. If a 3-vector is provided, it is interpreted such that its projection points upwards on the projected image seen by the camera. Thus, this 3-vector must *not* be parallel to the direction.
`fov`	field of view in degrees, as used in `view3d`. This is roughly the field-of-view seen along the shortest axis of the window.

Value

None

Author(s)

Danail Obreschkow

Examples

```
# Draw knot
rgl::open3d()
a = seq(0,2*pi,len=25)
knot = rgl::cylinder3d(center=cbind(sin(a)+2*sin(2*a), 2*sin(3*a), cos(a)-2*cos(2*a)),
                       e1 = cbind(cos(a)+4*cos(2*a), 6*cos(3*a), sin(a)+4*sin(2*a)),
                       radius = 0.8, closed = TRUE)
rgl::shade3d(rgl::addNormals(rgl::subdivision3d(knot,depth=2)), col="purple")

# Place static camera
rgl.camera(c(10,0,0),fov=50)

# Animate camera
## Not run:
  for(alpha in seq(0,2*pi,len=100)) {
    rgl.camera(10*c(cos(alpha),sin(alpha),0),fov=50)
  }

## End(Not run)
```

---

rgl.close.all *Close all open rgl windows*

Description

Checks if any rgl windows are currently open and, if so, closes them.

Usage

```
rgl.close.all()
```

Author(s)

Danail Obreschkow

rgl.draw *Continue drawing on screen*

Description

Updates screen display after this was stopped using the function rgl.hold. This is identical to calling par3d(skipRedraw=FALSE).

Usage

```
rgl.draw()
```

Value

None

Author(s)

Danail Obreschkow

See Also

rgl.hold

rgl.hold *Stop drawing on screen*

Description

Prevents the following rgl functions from drawing on the screen, until the function rgl.draw is called. This is used to accelerate complex drawings. This routine is identical to calling par3d(skipRedraw=TRUE).

Usage

```
rgl.hold()
```

Value

None

Author(s)

Danail Obreschkow

See Also

rgl.draw

`rgl.makemovie` *Produce a movie from and 3d rgl scene*

Description

Generates an MP4-movie of a 3d rgl scene with time-dependent objects and/or a camera path. The routine has been developed and tested for MacOS and it requires on a working installation of ffmpeg.

Usage

```
rgl.makemovie(
  frame = NULL,
  path = NULL,
  tmin = 0,
  tmax = 1,
  nframes = 60,
  fps = 60,
  output.path,
  output.filename,
  keep.frames = FALSE,
  quiet = TRUE,
  separator = .Platform$file.sep,
  ffmpeg.cmd = "ffmpeg",
  ffmpeg.opt = "-vcodec libx264 -crf 18 -pix_fmt yuv420p",
  manual = FALSE
)
```

Arguments

`frame`	optional function that plots or updates the 3D scene at a given time. This function must have exactly one argument, which specifies the time of the frame.
`path`	optional list that specifies the motion of the camera at some discrete times. The list contains the following elements (for more details see `rgl.camera`): `time` = optional n-vector of strictly monotonically increasing discrete times, required if and only if one of the following four arguments (position, direction, up, fov) are provided as matrices/vectors. If not given, equally spaced in times between tmin and tmax are assumed. `position` = optional argument specifying the camera position along the path. This argument must be one of three types: (1) A 3-element vector specifies a fixed camera position for the whole movie. (2) A n-by-3 matrix specifies n discrete camera positions at the exact times given in the `time` vector (see above). The code automatically generates a smooth function going through these n points. (3) A function f(t) of a single time variable t, which returns a 3-element vector, specifies the exact position at that time. `direction` = optional argument specifying the direction of the camera's optical

axis. This argument can be of the same three types as the `position` argument.
`up` = optional argument specifying the camera's up-direction. This argument can be of the same three types as the `position` argument.
`fov` = optional argument specifying the field-of-view (FoV) in degrees. Similarly to the above arguments, this can be either a single number (fixed FoV), a n-element vector (specifying the Fov at the n discrete times), or a scalar function (specifying the FoV at any time t).

`tmin`	physical time of first frame in the movie.
`tmax`	physical time of last frame in the movie.
`nframes`	number of frames in the movie. The time variable is sampled evenly between `tmin` and `tmax`.
`fps`	number of frames per second
`output.path`	character specifying the directory, where the movie and temporary frames are saved
`output.filename`	movie filename without path. This filename should end on the extension '.mp4'.
`keep.frames`	logical flag specifying whether the temporary directory with the individual frame files should be kept
`quiet`	logical flag; if true, all console outputs produced by 'ffmpeg' are suppressed
`separator`	filename separate of the system ('/' for Mac, Linux, Unix; '\' for Windows)
`ffmpeg.cmd`	command used to call ffmpeg form a terminal. Normally, this is just 'ffmpeg'.
`ffmpeg.opt`	optional arguments used with ffmpeg, such as compression and formatting options (see `https://www.ffmpeg.org/ffmpeg.html`).
`manual`	logical flag, if TRUE, ffmpeg is not run automatically. The ffmpeg command line is returned.

Details

Note that the frame width and height should be divisible by 2 for mp4 video compression to work. To accelerate the movie generation, it is possible to suppress the screen update by calling `rgl.hold` before calling `rgl.makemovie`.

Value

Returns the command line to run ffmpeg in a terminal.

Author(s)

Danail Obreschkow

Examples

```
rgl.new(aspect=4/3, col='black', xlim=c(-4,4), ylim=c(-4,4), zlim=c(-4,4))
rgl::clear3d(type = "lights")
rgl::light3d(30,60,viewpoint.rel = FALSE)


# Make frame function
frame = function(t) {
  # t = time in seconds
  rgl.hold()
  if (t>0) {for (i in seq(3)) rgl::pop3d()}
 rgl.ball(0, 0, 0, 1, normals='improved', depth=6, png=system.file('earth.png', package='rglplus'),
          emission='#444466', rotation=rgl::rotationMatrix(t/86400*2*pi,0,0,1))
  alpha = seq(0,2*pi,length=360)+2*pi*t/43200
  alpha = c(alpha[1],rep(alpha[2:359],each=2),alpha[360])
  y = 3.168*cos(alpha)
  z = 3.168*sin(alpha)
  rgl.ball(0,y[1],z[1],0.05,col='red',emission='#aa0000')
  rgl::segments3d(0,y,z,col='red',alpha=seq(0,1,length=720))
  rgl.draw()
}

# Make path
path = list(position=c(10,10,0), up=c(0,0.5,1), fov = function(t) 40-t/8640)

# Produce movie
## Not run:
rgl.makemovie(frame=frame, path=path, tmin=0, tmax=86400, output.path='~/testmovie',
              output.filename = 'movie.mp4', ffmpeg.cmd = 'ffmpeg', nframes=600)

## End(Not run)
```

`rgl.new` *Open and initialize new 3D plot*

Description

Calls open3d and various additional functions to initialize a 3d plot.

Usage

```
rgl.new(
  width = 0.5,
  aspect = 16/9,
  orientation = "xy",
  fov = 30,
```

```
  col = "white",
  light = TRUE,
  xlim = c(0, 1),
  ylim = c(0, 1),
  zlim = c(0, 1),
  xlab = NULL,
  ylab = NULL,
  zlab = NULL,
  axes = FALSE,
  fixed = TRUE,
  close.all = TRUE,
  ...
)
```

Arguments

width	either an integer (>1) specifying the number of pixels in the horizontal direction, or a real value (>0 and <=1) specifying the fraction of the available pixels. If the selected aspect ratio causes the number of vertical pixels to exceed the available number, the width is reduced as much as necessary.
aspect	aspect ratio of window, defined as the ratio of vertical-to-horizontal size.
orientation	3-by-3 rotation matrix or 2-character string specifying the orientation of the camera. For character string the allowed values are 'xy', 'yx', 'yz', 'zy', 'zx', 'xz', where the first letter is the axis displayed from left to right and the second letter is the axis displayed from bottom to top. The third axis points either out of the screen or into the screen following the right-hand convention. This is the same as the plane argument of rgl.orthoview.
fov	field of view in degrees, as used in view3d
col	background color
light	logical flag. If TRUE, the standard light source created by open3d will be light up the scene. If FALSE, no light source is added and the user must create custom light sources manually by calling light3d.
xlim	2-vector specifying the range along the x-axis
ylim	2-vector specifying the range along the y-axis
zlim	2-vector specifying the range along the z-axis
xlab	character string specifying the label of the x-axis
ylab	character string specifying the label of the y-axis
zlab	character string specifying the label of the z-axis
axes	logical flag specifying whether axes are displayed
fixed	logical flag. If TRUE (default), the range of the axes is *not* adjusted as objects are drawn.
close.all	logical flag. If TRUE (default), all existing rgl windows are closed before the new window is opened.
...	additional arguments for view3d.

Value

None

Author(s)

Danail Obreschkow

`rgl.orthoview` *Display orthogonal projection*

Description

Display orthogonal projection on principal Cartesian planes, with scene centre in the image centre.

Usage

```
rgl.orthoview(plane = "xy", fov = 0, ...)
```

Arguments

`plane`	character string, which can be either of 'xy', 'yx', 'yz', 'zy', 'zx', 'xz', where the first letter is the axis displayed from left to right and the second letter is the axis displayed from bottom to top. The third axis points either out of the screen or into the screen following the right-hand convention.
`fov`	field of view in degrees, as used in `view3d`.
`...`	additional arguments for `view3d`.

Value

None

Author(s)

Danail Obreschkow

See Also

`rgl.camera`

Examples

```
rgl::plot3d(array(runif(60),c(20,3)), col=rainbow(20), axes=FALSE,
            xlim=c(0,1), ylim=c(0,1), zlim=c(0,1), xlab='', ylab='', zlab='')
rgl::box3d()
rgl.orthoview('xy', fov=20)
```

rgl.test.scene *Plot 3D test image*

Description

Draws a c3D test image with three Cartesian axes, a sphere of radius 0.5 and three light sources with RGB colors.

Usage

```
rgl.test.scene(center = c(0, 0, 0), width = 0.5)
```

Arguments

center	3-vector specifying the centre of the 3D plot.
width	either an integer (>1) specifying the number of pixels in the horizontal direction, or a real value (>0 and <=1) specifying the fraction of the available pixels. If the selected aspect ratio causes the number of vertical pixels to exceed the available number, the width is reduced as much as necessary.

Value

None

Author(s)

Danail Obreschkow

rglplus *Extension of the 'rgl' 3D Visualization Package*

Description

Provides 3D plotting routines that facilitate the use of the 'rgl' package and extend its functionality. For example, the routines allow the user to directly control the camera position & orientation, as well as to generate 3D movies with a moving observer.

Author(s)

Danail Obreschkow

Index