Package 'MeshesTools'

October 29, 2022

Type Package

Title Some Tools for 3D Meshes

Version 1.0.0

Maintainer Stéphane Laurent <laurent_step@outlook.fr>

Description Provides some utilities for 3D meshes: clipping of a mesh to the volume bounded by another mesh, decomposition into convex parts, distance between a mesh and a point, volume, area, and centroid. All algorithms are performed by the 'C++' library 'CGAL' (<https://www.cgal.org/>).

License GPL-3

URL https://github.com/stla/MeshesTools

BugReports https://github.com/stla/MeshesTools/issues

Depends R (>= 2.10)

Imports data.table, gmp, PolygonSoup, Rcpp (>= 1.0.9), rgl, Rvcg

Suggests randomcoloR, rmarchingcubes

LinkingTo BH, Rcpp, RcppCGAL, RcppEigen

Encoding UTF-8

LazyData true

RoxygenNote 7.2.1

SystemRequirements C++ 17, gmp, mpfr

NeedsCompilation yes

Author Stéphane Laurent [aut, cre]

Repository CRAN

Date/Publication 2022-10-29 08:52:42 UTC

R topics documented:

`clipMesh` *Clip a mesh*

Description

Clip a mesh to the volume bounded by another mesh.

Usage

```
clipMesh(mesh, clipper, clipVolume = TRUE, normals = FALSE)
```

Arguments

`mesh`	a mesh given either as a list containing (at least) the fields `vertices` and `faces`, otherwise a **rgl** mesh (i.e. a `mesh3d` object)
`clipper`	a mesh given either as a list containing (at least) the fields `vertices` and `faces`, otherwise a **rgl** mesh (i.e. a `mesh3d` object)
`clipVolume`	Boolean, whether the clipping has to be done on the volume bounded by `mesh` rather than on its surface (i.e. `mesh` will be kept closed if it is closed)
`normals`	Boolean, whether to compute the vertex normals of the output mesh

Value

A triangle mesh of class `cgalMesh` (see `Mesh` for details).

Note

The clipping mesh (`clipper`) must be closed.

Examples

```
# cube clipped to sphere
library(MeshesTools)
library(rgl)
mesh    <- cube3d()
clipper <- sphereMesh(r= sqrt(2))
clippedMesh <- clipMesh(mesh, clipper)
open3d(windowRect = c(50, 50, 562, 562))
view3d(zoom = 0.9)
shade3d(toRGL(clippedMesh), color = "purple")

# Togliatti surface clipped to a ball ####
library(rmarchingcubes)
library(rgl)
library(MeshesTools)

# Togliatti surface equation: f(x,y,z) = 0
f <- function(x, y, z) {
  64*(x-1) *
    (x^4 - 4*x^3 - 10*x^2*y^2 - 4*x^2 + 16*x - 20*x*y^2 + 5*y^4 + 16 - 20*y^2) -
    5*sqrt(5-sqrt(5))*(2*z - sqrt(5-sqrt(5))) *
    (4*(x^2 + y^2 - z^2) + (1 + 3*sqrt(5)))^2
}

# grid
n <- 200L
x <- y <- seq(-5, 5, length.out = n)
z <- seq(-4, 4, length.out = n)
Grid <- expand.grid(X = x, Y = y, Z = z)
# calculate voxel
voxel <- array(with(Grid, f(X, Y, Z)), dim = c(n, n, n))
# calculate isosurface
contour_shape <- contour3d(
  griddata = voxel, level = 0, x = x, y = y, z = z
)
# make rgl mesh (plotted later)
mesh <- tmesh3d(
  vertices = t(contour_shape[["vertices"]]),
  indices  = t(contour_shape[["triangles"]]),
  normals  = contour_shape[["normals"]],
  homogeneous = FALSE
)

# clip to sphere of radius 4.8
clipper <- sphereMesh(r = 4.8)
clippedMesh <- clipMesh(mesh, clipper, clipVolume = FALSE, normals = TRUE)

# plot
open3d(windowRect = c(50, 50, 950, 500))
mfrow3d(1L, 2L)
view3d(0, -70, zoom = 0.8)
shade3d(mesh, color = "firebrick")
```

```
next3d()
view3d(0, -70, zoom = 0.8)
shade3d(toRGL(clippedMesh), color = "firebrick")
```

---

convexParts *Decomposition into convex parts*

Description

Decomposition of a mesh into convex parts.

Usage

```
convexParts(mesh, triangulate = TRUE)
```

Arguments

mesh	either a list containing the fields vertices and faces, otherwise a **rgl** mesh (i.e. a mesh3d object)
triangulate	Boolean, whether to triangulate the convex parts

Value

A list of cgalMesh lists (see Mesh), each corresponding to a convex part.

Examples

```
# a non-convex polyhedron ####
library(MeshesTools)
library(rgl)
library(randomcoloR)
meshes <- convexParts(mesh = NonConvexPolyhedron)
ncp <- length(meshes)
colors <- randomColor(ncp, hue = "random", luminosity = "bright")
open3d(windowRect = c(50, 50, 562, 562), zoom = 0.8)
for(i in seq_len(ncp)){
  shade3d(toRGL(meshes[[i]]), color = colors[i])
}

# pentagrammic prism ####
library(MeshesTools)
library(rgl)
library(randomcoloR)
data(pentagrammicPrism, package = "PolygonSoup")
meshes <- convexParts(mesh = pentagrammicPrism)
ncp <- length(meshes)
colors <- randomColor(ncp, hue = "random", luminosity = "bright")
open3d(windowRect = c(50, 50, 562, 562), zoom = 0.8)
for(i in seq_len(ncp)){
  shade3d(toRGL(meshes[[i]]), color = colors[i])
}
```

`cyclideMesh` *Cyclide mesh*

Description

Triangle mesh of a Dupin cyclide.

Usage

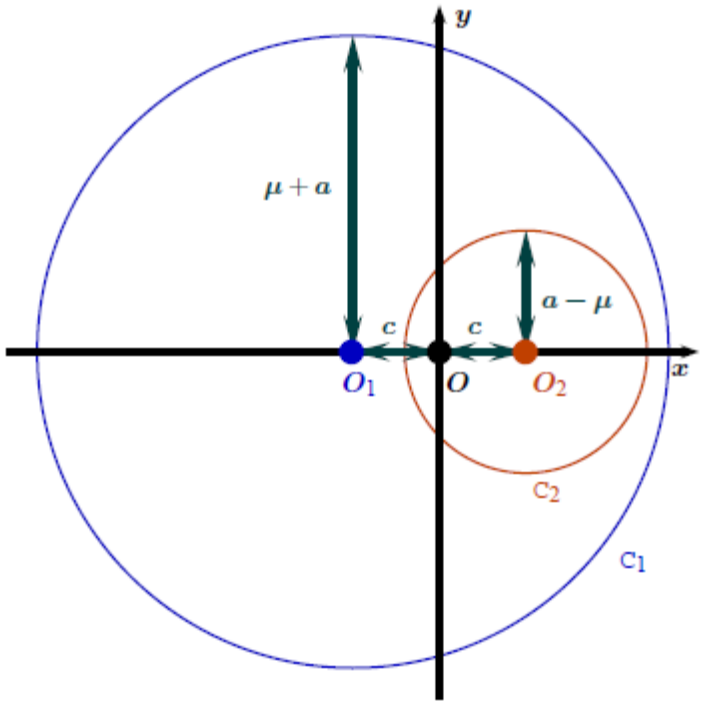
```
cyclideMesh(a, c, mu, nu = 90L, nv = 40L)
```

Arguments

`a, c, mu`	cyclide parameters, positive numbers such that `c < mu < a`
`nu, nv`	numbers of subdivisions, integers (at least 3)

Details

The Dupin cyclide in the plane *z=0*:



Value

A triangle **rgl** mesh (class `mesh3d`).

Examples

```
library(MeshesTools)
library(rgl)
mesh <- cyclideMesh(a = 97, c = 32, mu = 57)
sphere <- sphereMesh(x = 32, y = 0, z = 0, r = 40)
open3d(windowRect = c(50, 50, 562, 562))
view3d(0, 0, zoom = 0.75)
shade3d(mesh, color = "chartreuse")
wire3d(mesh)
shade3d(sphere, color = "red")
wire3d(sphere)
```

---

distancesToMesh *Distance to a mesh*

Description

Computes the distances from given points to a mesh.

Usage

```
distancesToMesh(mesh, points)
```

Arguments

`mesh`	a mesh given either as a list containing (at least) the fields `vertices` and `faces`, otherwise a **rgl** mesh (i.e. a `mesh3d` object)
`points`	either one point given as a numeric vector or several points given as a numeric matrix with three columns

Value

A numeric vector providing the distances between the given point(s) to the mesh.

Examples

```
# cube example ####
library(MeshesTools)
mesh <- rgl::cube3d()
points <- rbind(
  c(0, 0, 0),
  c(1, 1, 1)
)
distancesToMesh(mesh, points) # should be 1 and 0

# cyclide example ####
library(MeshesTools)
a <- 100; c <- 30; mu <- 80
mesh <- cyclideMesh(a, c, mu, nu = 100L, nv = 100L)
```

```
O2 <- c(c, 0, 0)
# should be a - mu = 20 (see ?cyclideMesh):
distancesToMesh(mesh, O2)
```

`greatStellatedDodecahedron` *Great stellated dodecahedron*

Description

A list representing the great stellated dodecahedron. It has 32 vertices, 60 triangular faces and 90 edges.

Usage

```
greatStellatedDodecahedron
```

Format

A list with fields `vertices`, `faces` and `edges`.

`HopfTorusMesh` *Hopf torus mesh*

Description

Triangle mesh of a Hopf torus.

Usage

```
HopfTorusMesh(nlobes = 3, A = 0.44, alpha = NULL, nu, nv)
```

Arguments

`nlobes`	number of lobes of the Hopf torus, a positive integr
`A`	parameter of the Hopf torus, number strictly between `0` and `pi/2`
`alpha`	if not `NULL`, this is the exponent of a modified stereographic projection, a positive number; otherwise the ordinary stereographic projection is used
`nu, nv`	numbers of subdivisions, integers (at least 3)

Value

A triangle **rgl** mesh (class `mesh3d`).

Examples

```
library(MeshesTools)
library(rgl)
mesh <- HopfTorusMesh(nu = 90, nv = 90)
open3d(windowRect = c(50, 50, 562, 562))
view3d(0, 0, zoom = 0.75)
shade3d(mesh, color = "forestgreen")
wire3d(mesh)
mesh <- HopfTorusMesh(nu = 90, nv = 90, alpha = 1.5)
open3d(windowRect = c(50, 50, 562, 562))
view3d(0, 0, zoom = 0.75)
shade3d(mesh, color = "yellowgreen")
wire3d(mesh)
```

---

meshArea *Mesh area*

Description

Computes the surface area a mesh.

Usage

```
meshArea(mesh)
```

Arguments

`mesh`	a mesh given either as a list containing (at least) the two fields `vertices` (numeric matrix with three columns) and `faces` (integer matrix or list of integer vectors), otherwise as a **rgl** mesh (i.e. a `mesh3d` object)

Value

A number, the surface area of the mesh.

Examples

```
library(MeshesTools)
R <- 4; r <- 2
mesh <- torusMesh(R, r)
meshArea(mesh)
# true area of the torus:
4 * pi^2 * R * r
```

meshCentroid *Mesh centroid*

Description

Computes the centroid of a closed mesh.

Usage

```
meshCentroid(mesh)
```

Arguments

mesh	a mesh given either as a list containing (at least) the two fields `vertices` (numeric matrix with three columns) and `faces` (integer matrix or list of integer vectors), otherwise as a **rgl** mesh (i.e. a `mesh3d` object)

Value

The centroid of the mesh given as a numeric vector.

Examples

```
library(MeshesTools)
mesh <- cyclideMesh(a = 97, c = 32, mu = 57)
meshCentroid(mesh)
```

MeshesTools-imports *Objects imported from other packages*

Description

These objects are imported from other packages. Follow the links to their documentation: toRGL, plotEdges.

meshVolume *Mesh volume*

Description

Computes the volume bounded by a mesh.

Usage

```
meshVolume(mesh)
```

Arguments

mesh	a mesh given either as a list containing (at least) the two fields `vertices` (numeric matrix with three columns) and `faces` (integer matrix or list of integer vectors), otherwise as a **rgl** mesh (i.e. a `mesh3d` object)

Value

A number, the volume bounded by the mesh.

Examples

```
library(MeshesTools)
R <- 4; r <- 2
mesh <- torusMesh(R, r)
meshVolume(mesh)
# true volume of the torus:
2 * pi^2 * R * r^2
```

NonConvexPolyhedron *A mesh of a non-convex polyhedron*

Description

A list representing a non-convex polyhedron with 14 vertices and 24 triangular faces.

Usage

```
NonConvexPolyhedron
```

Format

A list with fields `vertices` and `faces`.

sphereMesh *Sphere mesh*

Description

Mesh of a sphere.

Usage

```
sphereMesh(x = 0, y = 0, z = 0, r = 1, iterations = 3L)
```

Arguments

x, y, z	coordinates of the center
r	radius
iterations	number of iterations

Value

A **rgl** mesh (class mesh3d).

torusMesh *Torus mesh*

Description

Triangle mesh of a torus.

Usage

```
torusMesh(R, r, nu = 50, nv = 30)
```

Arguments

R, r	major and minor radii, positive numbers
nu, nv	numbers of subdivisions, integers (at least 3)

Value

A triangle **rgl** mesh (class mesh3d).

Examples

```
library(MeshesTools)
library(rgl)
mesh <- torusMesh(R = 3, r = 1)
open3d(windowRect = c(50, 50, 562, 562))
view3d(0, 0, zoom = 0.75)
shade3d(mesh, color = "green")
wire3d(mesh)
```

Index