Package 'PolygonSoup'

October 19, 2022

Type Package

Title Mesh from Polygon Soup

Version 1.0.1

Maintainer Stéphane Laurent <laurent_step@outlook.fr>

Description Allows to get a consistent 3D mesh from a polygon soup, that is an unorganized set of polygons. The mesh can be triangulated and its exterior edges are computed.

License GPL-3

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

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

Depends R (>= 2.10)

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

Suggests misc3d

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 2022-10-19 13:37:56 UTC

R topics documented:

`Mesh` *Make a 3D mesh*

Description

Make a 3D mesh from given vertices and faces; the returned faces are coherently oriented, normals are computed if desired, and triangulation is performed if desired. The mesh is also cleaned: duplicated vertices or faces are merged, and isolated vertices are removed.

Usage

```
Mesh(vertices, faces, mesh = NULL, triangulate = FALSE, normals = FALSE)
```

Arguments

`vertices`	a numeric matrix with three columns, or a `bigq` matrix with three columns
`faces`	either an integer matrix (each row provides the vertex indices of the corresponding face) or a list of integer vectors, each one providing the vertex indices of the corresponding face
`mesh`	if not `NULL`, this argument takes precedence over `vertices` and `faces`, and must be either a list containing the fields `vertices` and `faces` (objects as described above), otherwise a **rgl** mesh (i.e. a `mesh3d` object)
`triangulate`	Boolean, whether to triangulate the faces
`normals`	Boolean, whether to compute the normals

Value

A list giving the vertices, the edges, the faces of the mesh, the exterior edges, the exterior vertices and optionally the normals. If `triangulate=TRUE`, this list has two additional components `edges0` and `normals0` giving the edges and the normals before the triangulation, unless the mesh is already triangulated, in which case the `triangulate` option is ignored.

See Also

See `plotEdges` for more details about the edges returned by this function.

Examples

```
library(PolygonSoup)
library(rgl)

# a tetrahedron with ill-oriented faces ####
vertices <- rbind(
  c(-1, -1, -1),
  c(1, 1, -1),
  c(1, -1, 1),
  c(-1, 1, 1)
)
faces <- rbind(
  c(1, 2, 3),
  c(3, 4, 2),
  c(4, 2, 1),
  c(4, 3, 1)
)

# plot the tetrahedron, hiding the back of the faces
# then some faces do not appear, as their orientation is not correct
tmesh1 <- tmesh3d(
  vertices = t(vertices),
  indices = t(faces),
  homogeneous = FALSE
)
open3d(windowRect = c(50, 50, 562, 562))
shade3d(tmesh1, color = "green", back = "cull")

# now run the `Mesh` function
mesh2 <- Mesh(vertices, faces, normals = FALSE)
# plot the tetrahedron, hiding the back of the faces
# then all faces appear now
tmesh2 <- toRGL(mesh2)
open3d(windowRect = c(50, 50, 562, 562))
shade3d(tmesh2, color = "blue", back = "cull")

# illustration of the cleaning feature ####
# we construct a mesh with a lot of duplicated vertices
library(misc3d) # to compute a mesh of an isosurface
a <- 0.94; mu <- 0.56; c <- 0.34 # cyclide parameters
f <- function(x, y, z, a, c, mu){ # implicit equation of the cyclide
  b <- sqrt(a^2 - c^2)
  (x^2 + y^2 + z^2 - mu^2 + b^2)^2 - 4*(a*x - c*mu)^2 - 4*b^2*y^2
}
x <- seq(-c - mu - a, abs(mu - c) + a, length.out = 45)
y <- seq(-mu - a, mu + a, length.out = 45)
z <- seq(-mu - c, mu + c, length.out = 30)
g <- expand.grid(x = x, y = y, z = z)
voxel <- array(with(g, f(x, y, z, a, c, mu)), c(45, 45, 30))
cont <- computeContour3d(voxel, level = 0, x = x, y = y, z = z)
ids <- matrix(1:nrow(cont), ncol = 3, byrow = TRUE)
# run the `Mesh` function
```

```
mesh <- Mesh(cont, ids, normals = TRUE)
# plot the cyclide
tmesh <- toRGL(mesh)
open3d(windowRect = c(50, 50, 562, 562), zoom = 0.9)
shade3d(tmesh, color = "green")

# illustration of the `triangulate` option ####
# the faces of the truncated icosahedron are hexagonal or pentagonal:
truncatedIcosahedron[["faces"]]
# so we triangulate them:
mesh <- Mesh(
  mesh = truncatedIcosahedron,
  triangulate = TRUE, normals = FALSE
)
# now we can plot the truncated icosahedron
tmesh <- toRGL(mesh)
open3d(windowRect = c(50, 50, 562, 562), zoom = 0.9)
shade3d(tmesh, color = "orange")
```

`pentagrammicPrism` *A mesh of a pentagrammic prism*

Description

A list representing a pentagrammic prism, giving the vertices and the faces; it has 20 vertices, 10 triangular faces, 10 rectangular faces and two pentagonal faces.

Usage

```
pentagrammicPrism
```

Format

A list (`vertices`, `faces`).

`plotEdges` *Plot some edges*

Description

Plot the given edges with **rgl**.

Usage

```
plotEdges(
  vertices,
  edges,
  color = "black",
  lwd = 2,
  edgesAsTubes = TRUE,
  tubesRadius = 0.03,
  verticesAsSpheres = TRUE,
  only = NULL,
  spheresRadius = 0.05,
  spheresColor = color
)
```

Arguments

vertices	a three-columns matrix giving the coordinates of the vertices
edges	a two-columns integer matrix giving the edges by pairs of vertex indices
color	a color for the edges
lwd	line width, a positive number, ignored if edgesAsTubes=TRUE
edgesAsTubes	Boolean, whether to draw the edges as tubes
tubesRadius	the radius of the tubes when edgesAsTubes=TRUE
verticesAsSpheres	Boolean, whether to draw the vertices as spheres
only	integer vector made of the indices of the vertices you want to plot (as spheres), or NULL to plot all vertices
spheresRadius	the radius of the spheres when verticesAsSpheres=TRUE
spheresColor	the color of the spheres when verticesAsSpheres=TRUE

Value

No value.

Examples

```
library(PolygonSoup)
library(rgl)

# we triangulate the truncated icosahedron mesh
mesh <- Mesh(
  mesh = truncatedIcosahedron,
  triangulate = TRUE, normals = FALSE
)
# now we can plot the truncated icosahedron
tmesh <- toRGL(mesh)
open3d(windowRect = c(50, 50, 562, 562), zoom = 0.9)
shade3d(tmesh, color = "gold")
```

```
# we plot the edges given in `mesh[["edges0"]]`; these are the
# edges of the mesh before the triangulation
plotEdges(mesh[["vertices"]], mesh[["edges0"]], color = "navy")

# we triangulate the pentagrammic prism mesh
mesh <- Mesh(
  mesh = pentagrammicPrism,
  triangulate = TRUE, normals = FALSE
)
# now we can plot the pentagrammic prism
tmesh <- toRGL(mesh)
open3d(windowRect = c(50, 50, 562, 562), zoom = 0.9)
shade3d(tmesh, color = "navy")
# we plot the exterior edges only, given in `mesh[["exteriorEdges"]]`
plotEdges(
  mesh[["vertices"]], mesh[["exteriorEdges"]], color = "gold",
  tubesRadius = 0.02, spheresRadius = 0.02
)

# or only plot the edges whose corresponding dihedral angle is acute:
allEdges <- mesh[["edgesDF"]]
edges <- as.matrix(subset(allEdges, angle <= 91, select = c("i1", "i2")))
```

`readMeshFile` *Read a mesh file*

Description

Read mesh vertices and faces from a file.

Usage

```
readMeshFile(filepath)
```

Arguments

`filepath`	path to the mesh file; supported formats are `stl`, `ply`, `obj` and `off`

Value

A list with two fields: `vertices`, a numeric matrix with three columns, and `faces`, either a list of integer vectors or, in the case if all faces have the same number of sides, an integer matrix.

Examples

```
library(PolygonSoup)
library(rgl)
vf <- readMeshFile(
  system.file("extdata", "beethoven.ply", package = "PolygonSoup")
)
```

```
mesh <- Mesh(
  vf[["vertices"]], vf[["faces"]], normals = TRUE
)
rglmesh <- toRGL(mesh)
open3d(windowRect = c(50, 50, 562, 562))
view3d(0, 0, zoom = 0.8)
shade3d(rglmesh, color = "palevioletred")
```

---

toRGL *Conversion to 'rgl' mesh*

Description

Converts a CGAL mesh (e.g. an output of the `Mesh` function) to a **rgl** mesh.

Usage

```
toRGL(mesh, ...)
```

Arguments

`mesh`	a CGAL mesh, that is to say a list of class `"cgalMesh"` (e.g. an output of the `Mesh` function); in order to be convertible to a **rgl** mesh, its faces must have at most four sides
`...`	arguments passed to `mesh3d`

Value

A **rgl** mesh, that is to say a list of class `"mesh3d"`.

Examples

```
library(PolygonSoup)
library(rgl)
mesh <- Mesh(
  truncatedIcosahedron[["vertices"]], truncatedIcosahedron[["faces"]],
  triangulate = TRUE
)
rglmesh <- toRGL(mesh, segments = t(mesh[["edges"]]))
open3d(windowRect = c(50, 50, 562, 562), zoom = 0.9)
shade3d(rglmesh, color = "darkred")
```

truncatedIcosahedron *A mesh of the truncated icosahedron*

Description

A list giving the vertices and the faces of a truncated icosahedron. There are some hexagonal faces and some pentagonal faces.

Usage

```
truncatedIcosahedron
```

Format

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

writeMeshFile *Export mesh to a file*

Description

Export a mesh to a file.

Usage

```
writeMeshFile(mesh, filename, precision = 17L, binary = 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)
`filename`	name of the file to be written, with extension `stl`, `ply`, `obj` or `off`
`precision`	positive integer, number of decimal digits for the vertices
`binary`	Boolean, whether to write a binary file or an ASCII file

Value

No value, just generates the file.

Index