Package 'Boov'

October 31, 2022

Type Package

Title Boolean Operations on Volumes

Version 1.0.0

Maintainer Stéphane Laurent <laurent_step@outlook.fr>

Description Performs Boolean operations on volumes: union, difference and intersection. The computations are done by the 'C++' library 'CGAL' <https://www.cgal.org/>.

License GPL-3

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

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

Depends R (>= 2.10)

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

Suggests rgl

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 2022-10-31 12:52:40 UTC

R topics documented:

`Boov-imports` *Objects imported from other packages*

Description

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

`MeshesDifference` *Meshes difference*

Description

Computes the difference between two meshes.

Usage

```
MeshesDifference(mesh1, mesh2, clean = TRUE, normals = FALSE)
```

Arguments

`mesh1, mesh2`	two meshes, each being given as either a **rgl** mesh, or a list with (at least) two fields: `vertices` and `faces`; the `vertices` matrix can be a numeric matrix or a matrix of `bigq` rational numbers (from the **gmp** package)
`clean`	Boolean, whether to clean the meshes (merging duplicated vertices, duplicated faces, removing isolated vertices); set to `FALSE` if you know your meshes are clean
`normals`	Boolean, whether to return the per-vertex normals of the output mesh

Value

A triangle mesh given as a list with fields `vertices`, `faces`, `edges`, `exteriorEdges`, `gmpvertices` if using **gmp** meshes, and `normals` if `normals=TRUE`.

Examples

```
library(Boov)
library(rgl)

# mesh one: a cube
mesh1 <- cube3d() # (from the rgl package)

# mesh two: another cube
mesh2 <- translate3d( # (from the rgl package)
  cube3d(), 1, 1, 0
)

# compute the difference
differ <- MeshesDifference(mesh1, mesh2)

# plot
rgldiffer <- toRGL(differ)
open3d(windowRect = c(50, 50, 562, 562))
shade3d(mesh1, color = "yellow", alpha = 0.2)
shade3d(mesh2, color = "cyan", alpha = 0.2)
shade3d(rgldiffer, color = "red")
plotEdges(
  vertices = differ[["vertices"]], edges = differ[["exteriorEdges"]],
  edgesAsTubes = TRUE, verticesAsSpheres = TRUE
)
```

`MeshesIntersection` *Meshes intersection*

Description

Computes the intersection of the given meshes.

Usage

```
MeshesIntersection(meshes, clean = TRUE, normals = FALSE)
```

Arguments

`meshes`	a list of meshes, each being either a **rgl** mesh, or as a list with (at least) two fields: `vertices` and `faces`; the `vertices` matrix can be a numeric matrix or a matrix of `bigq` rational numbers (from the **gmp** package)
`clean`	Boolean, whether to clean the meshes (merging duplicated vertices, duplicated faces, removing isolated vertices); set to `FALSE` if you are sure your meshes are clean, to gain some speed
`normals`	Boolean, whether to return the per-vertex normals of the output mesh

Value

A triangle mesh given as a list with fields `vertices`, `faces`, `edges`, `exteriorEdges`, `gmpvertices` if using **gmp** meshes, and `normals` if `normals=TRUE`.

Examples

```
library(Boov)
library(rgl)

# mesh one: truncated icosahedron; we triangulate it for plotting
library(PolygonSoup)
mesh1 <- Mesh(
  mesh = truncatedIcosahedron,
  triangulate = TRUE, normals = FALSE
)

# mesh two: a cube
mesh2 <- translate3d( # (from the rgl package)
  cube3d(), 2, 0, 0
)

# compute the intersection
inter <- MeshesIntersection(list(mesh1, mesh2))

# plot
rglmesh1 <- toRGL(mesh1)
rglinter <- toRGL(inter)
open3d(windowRect = c(50, 50, 562, 562))
shade3d(rglmesh1, color = "yellow", alpha = 0.2)
shade3d(mesh2, color = "cyan", alpha = 0.2)
shade3d(rglinter, color = "red")
plotEdges(
  vertices = inter[["vertices"]], edges = inter[["exteriorEdges"]],
  edgesAsTubes = FALSE, lwd = 3, verticesAsSpheres = FALSE
)

# other example, with 'gmp' rational numbers ####
library(Boov)
library(gmp)
library(rgl)

cube <- cube3d()

rglmesh1 <- cube
mesh1 <-
  list(vertices = t(cube[["vb"]][-4L, ]), faces = t(cube[["ib"]]))
mesh1[["vertices"]] <- as.bigq(mesh1[["vertices"]])

rotMatrix <- t(cbind( # pi/3 around a great diagonal
  as.bigq(c(2, -1, 2), c(3, 3, 3)),
  as.bigq(c(2, 2, -1), c(3, 3, 3)),
  as.bigq(c(-1, 2, 2), c(3, 3, 3))
```

```
))
mesh2 <-
  list(vertices = t(cube[["vb"]][-4L, ]), faces = t(cube[["ib"]]))
mesh2[["vertices"]] <- as.bigq(mesh2[["vertices"]]) %*% rotMatrix
rglmesh2 <- rotate3d(cube, pi/3, 1, 1, 1)

inter <- MeshesIntersection(list(mesh1, mesh2))
# perfect vertices:
inter[["gmpVertices"]]
rglinter <- toRGL(inter)

open3d(windowRect = c(50, 50, 562, 562), zoom = 0.9)
bg3d("#363940")
shade3d(rglmesh1, color = "yellow", alpha = 0.2)
shade3d(rglmesh2, color = "orange", alpha = 0.2)
shade3d(rglinter, color = "hotpink")
plotEdges(
  inter[["vertices"]], inter[["exteriorEdges"]],
  only = inter[["exteriorVertices"]],
  color = "firebrick",
  tubesRadius = 0.05, spheresRadius = 0.07
)
```

`MeshesUnion` *Meshes union*

Description

Computes the union of the given meshes.

Usage

```
MeshesUnion(meshes, clean = TRUE, normals = FALSE)
```

Arguments

`meshes`	a list of meshes, each being either a **rgl** mesh, or as a list with (at least) two fields: `vertices` and `faces`; the `vertices` matrix can be a numeric matrix or a matrix of `bigq` rational numbers (from the **gmp** package)
`clean`	Boolean, whether to clean the meshes (merging duplicated vertices, duplicated faces, removing isolated vertices); set to `FALSE` if you are sure your meshes are clean, to gain some speed
`normals`	Boolean, whether to return the per-vertex normals of the output mesh

Value

A triangle mesh given as a list with fields `vertices`, `faces`, `edges`, `exteriorEdges`, `gmpvertices` if using **gmp** meshes, and `normals` if `normals=TRUE`.

Examples

```
library(Boov)
library(rgl)

# mesh one: a cube
mesh1 <- cube3d() # (from the rgl package)

# mesh two: another cube
mesh2 <- translate3d( # (from the rgl package)
  cube3d(), 1, 1, 1
)

# compute the union
umesh <- MeshesUnion(list(mesh1, mesh2))

# plot
rglumesh <- toRGL(umesh)
open3d(windowRect = c(50, 50, 562, 562))
shade3d(rglumesh, color = "red")
plotEdges(
  vertices = umesh[["vertices"]], edges = umesh[["exteriorEdges"]],
  edgesAsTubes = TRUE, verticesAsSpheres = TRUE
)
```

---

octahedraCompound *Compound of five octahedra*

Description

Five octahedra in a pretty configuration. Each octahedron is centered at the origin.

Usage

```
octahedraCompound
```

Format

A list with three fields: the field `meshes` is a list of five elements, each one representing an octahedron by a list with two elements, the vertices and the faces; the field `rglmeshes` is the list of the five corresponding **rgl** meshes; the field `gmpmeshes` is the same as `meshes` except that the vertices are **gmp** rational numbers.

qsqrt *Rational approximation of square roots*

Description

Returns a rational approximation of the square root of an integer.

Usage

```
qsqrt(x, n)

qsqrt2(n)

qsqrt3(n)

qsqrtPhi(n)

## S3 method for class 'qsqrt'
print(x, ...)
```

Arguments

x	the positive integer whose square root is desired
n	a positive integer, the higher the better approximation
...	ignored

Value

The qsqrt function returns a **gmp** rational number (class bigq) approximating the square root of x. The qsqrt2, qsqrt3, and qsqrtPhi functions return a **gmp** rational number approximating the square root of 2, 3, and phi (the golden number) respectively. Their value converge more fastly than the value obtained with qsqrt.

Examples

```
library(Boov)
qsqrt(2, 7)
qsqrt2(7)
qsqrt3(22)
qsqrtPhi(17)
```

`tetrahedraCompound` *Compound of five tetrahedra*

Description

Five tetrahedra in a pretty configuration. Each tetrahedron is centered at the origin.

Usage

```
tetrahedraCompound
```

Format

A list with three fields: the field `meshes` is a list of five elements, each one representing a tetrahedron by a list with two elements, the vertices and the faces; the field `rglmeshes` is the list of the five corresponding **rgl** meshes; the field `gmpmeshes` is the same as `meshes` except that the vertices are **gmp** rational numbers.

Index