Package: alcyon (via r-universe)

October 28, 2024

Type Package

Title Spatial Network Analysis

Version 0.5.0

Description Interface package for 'sala', the spatial network analysis library from the 'depthmapX' software application. The R parts of the code are based on the 'rdepthmap' package. Allows for the analysis of urban and building-scale networks and provides metrics and methods usually found within the Space Syntax domain. Methods in this package are described by K. Al-Sayed, A. Turner, B. Hillier, S. Iida and A. Penn (2014) ``Space Syntax methodology'', and also by A. Turner (2004) <https://discovery.ucl.ac.uk/id/eprint/2651> ``Depthmap 4: a researcher's handbook''.

License GPL-3

Encoding UTF-8

LazyData true

LinkingTo Rcpp, cli

Imports Rcpp, methods

Depends sf, stars

Suggests knitr, rmarkdown, testthat

SystemRequirements C++17

VignetteBuilder knitr

Collate 'generics.R' 'rcppRoxygen.R' 'helper-loadMapsRoxygen.R' 'helper-processResult.R' 'ShapeMap.R' 'AxialShapeGraph.R' 'AllLineShapeGraph.R' 'SegmentShapeGraph.R' 'PointMap.R' 'TraversalType.R' 'AgentLookMode.R' 'RcppExports.R' 'agentAnalysis.R' 'allFewestLineMap.R' 'axialAnalysis.R' 'generateRandomCapString.R' 'getTopFeatures.R' 'isovist.R' 'matchPointsToLines.R' 'palettes.R' 'prepareVGA.R' 'readMetaGraph.R' 'refIDtoIndex.R' 'segmentAnalysis.R' 'sfConversions.R' 'shapegraphToGraphData.R' 'oneToOneTraverse.R' 'oneToAllTraverse.R' 'allToAllTraverse.R' 'vgaLocal.R'

RoxygenNote 7.3.2

Language en-GB

URL https://github.com/spatialnous/alcyon,
https://spatialnous.github.io/alcyon/

Repository https://spatialnous.r-universe.dev

RemoteUrl https://github.com/spatialnous/alcyon

RemoteRef HEAD

RemoteSha 496da21d98a0b06ab8b6ef13017bfdfad3339bfb

Contents

`agentAnalysis` *Agent Analysis*

Description

Runs Agent Analysis on the given PointMap

Usage

```
agentAnalysis(
  pointMap,
  timesteps,
  releaseRate,
  agentLifeTimesteps,
```

```
  agentFov,
  agentStepsToDecision,
  agentLookMode,
  originX = vector(),
  originY = vector(),
  locationSeed = 0L,
  numberOfTrails = 0L,
  getGateCounts = FALSE,
  copyMap = TRUE,
  verbose = FALSE,
  progress = FALSE
)
```

Arguments

`pointMap`	A PointMap, used as an exosomatic visual map for agents to take exploratory information
`timesteps`	Number of total system timesteps.
`releaseRate`	Agent release rate (likelihood of release per timestep).
`agentLifeTimesteps`	Agent total lifetime (in timesteps)
`agentFov`	Agent field-of-view (out of 32 bins = 360).
`agentStepsToDecision`	Agent steps before turn decision.
`agentLookMode`	The agent look mode. See AgentLookMode
`originX`	Agent starting points (x coordinates).
`originY`	Agent starting point (y coordinates).
`locationSeed`	Agents to start at random locations with specific seed (0 to 10). Default is 0.
`numberOfTrails`	Record trails for this amount of agents (set to 0 to record all, with max possible currently = 50).
`getGateCounts`	Get values at gates
`copyMap`	Optional. Copy the internal sala map
`verbose`	Optional. Show more information of the process.
`progress`	Optional. Show process progress.

Value

Returns a list with:

- newAttributes: The new attributes that were created during the process
- trailMap: A ShapeMap with trails if numberOfTrails was set over 0

Examples

```
mifFile <- system.file(
    "extdata", "testdata", "simple",
    "simple_interior.mif",
    package = "alcyon"
  )
  sfMap <- st_read(mifFile,
    geometry_column = 1L, quiet = TRUE
  )
  pointMap <- makeVGAPointMap(
    sfMap,
    gridSize = 0.5,
    fillX = 3.0,
    fillY = 6.0,
    maxVisibility = NA,
    boundaryGraph = FALSE,
    verbose = FALSE
  )
agentAnalysis(
  pointMap,
  timesteps = 3000L,
  releaseRate = 0.1,
  agentStepsToDecision = 3L,
  agentFov = 11L,
  agentLife = 1000L,
  agentLookMode = AgentLookMode$Standard,
  originX = NA,
  originY = NA,
  locationSeed = 1L,
  numberOfTrails = 50L,
  getGateCounts = FALSE,
  verbose = FALSE
)
```

AgentLookMode *Agent look modes.*

Description

These are meant to be used to indicate what kind of look function the agents use to look around and decide where to go next. Possible values:

- AgentLookMode$None
- AgentLookMode$Standard
- AgentLookMode$LineOfSightLength
- AgentLookMode$OcclusionLength
- AgentLookMode$OcclusionAny
- AgentLookMode$OcclusionGroup45 (Occlusion group bins - 45 degrees)

- AgentLookMode$OcclusionGroup60 (Occlusion group bins - 60 degrees)
- AgentLookMode$OcclusionFurthest (Furthest occlusion per bin)
- AgentLookMode$BinFarDistance (Per bin far distance weighted)
- AgentLookMode$BinAngle (Per bin angle weighted)
- AgentLookMode$BinFarDistanceAngle (Per bin far-distance and angle weighted)
- AgentLookMode$BinMemory (Per bin memory)

Usage

```
AgentLookMode
```

Format

An object of class `list` of length 12.

Value

A list of numbers representing each agent look mode

Examples

```
AgentLookMode$Standard
AgentLookMode$LineOfSightLength
AgentLookMode$OcclusionAny
```

AllLineShapeGraph-class

All-line Axial ShapeGraph

Description

A representation of sala's All-line ShapeGraph in R. Holds onto a sala All-line ShapeGraph pointer and operates on that

`allToAllTraverse` *All-to-all traversal*

Description

Runs all-to-all traversal on a map with a graph. This is applicable to:

- PointMaps (Visibility Graph Analysis)
- Axial ShapeGraphs (Axial analysis)
- Segment ShapeGraphs (Segment analysis)

Usage

```
allToAllTraverse(
  map,
  traversalType,
  radii,
  radiusTraversalType,
  weightByAttribute = NULL,
  includeBetweenness = FALSE,
  quantizationWidth = NA,
  gatesOnly = FALSE,
  nthreads = 1L,
  copyMap = TRUE,
  verbose = FALSE,
  progress = FALSE
)
```

Arguments

`map`	A PointMap, Axial ShapeGraph or Segment ShapeGraph
`traversalType`	The traversal type. See TraversalType
`radii`	A list of radii
`radiusTraversalType`	The traversal type to keep track of whether the analysis is within the each radius limit. See TraversalType
`weightByAttribute`	The attribute to weigh the analysis with
`includeBetweenness`	Set to TRUE to also calculate betweenness (known as Choice in the Space Syntax domain)
`quantizationWidth`	Set this to use chunks of this width instead of continuous values for the cost of traversal. This is equivalent to the "tulip bins" for depthmapX's tulip analysis (1024 tulip bins = pi/1024 quantizationWidth). Only works for Segment ShapeGraphs

gatesOnly	Optional. Only calculate results at particular gate pixels. Only works for PointMaps
nthreads	Optional. Use more than one threads. 1 by default, set to 0 to use all available. Only available for PointMaps.
copyMap	Optional. Copy the internal sala map
verbose	Optional. Show more information of the process.
progress	Optional. Enable progress display

Value

A new map with the results included

Examples

```
# Pointmap analysis (VGA)
mifFile <- system.file(
    "extdata", "testdata", "simple",
    "simple_interior.mif",
    package = "alcyon"
  )
  sfMap <- st_read(mifFile,
    geometry_column = 1L, quiet = TRUE
  )
  pointMap <- makeVGAPointMap(
    sfMap,
    gridSize = 0.5,
    fillX = 3.0,
    fillY = 6.0,
    maxVisibility = NA,
    boundaryGraph = FALSE,
    verbose = FALSE
  )
allToAllTraverse(pointMap,
  traversalType = TraversalType$Angular,
  radii = -1L,
  radiusTraversalType = TraversalType$None
)

# Axial analysis
mifFile <- system.file(
    "extdata", "testdata", "barnsbury",
    "barnsbury_small_axial_original.mif",
    package = "alcyon"
  )
  sfMap <- st_read(mifFile,
    geometry_column = 1L, quiet = TRUE
  )
  shapeGraph <- as(sfMap, "AxialShapeGraph")
allToAllTraverse(
  shapeGraph,
  traversalType = TraversalType$Topological,
  radii = c("n", "3"),
```

```
  includeBetweenness = TRUE
)

# Segment analysis
mifFile <- system.file(
    "extdata", "testdata", "barnsbury",
    "barnsbury_small_segment_original.mif",
    package = "alcyon"
  )
  sfMap <- st_read(mifFile,
    geometry_column = 1L, quiet = TRUE
  )
  shapeGraph <- as(sfMap, "SegmentShapeGraph")
allToAllTraverse(
  shapeGraph,
  radii = c("n", "100"),
  radiusTraversalType = TraversalType$Metric,
  traversalType = TraversalType$Angular,
  weightByAttribute = "Segment Length",
  includeBetweenness = TRUE,
  quantizationWidth = pi / 1024L,
  verbose = FALSE,
  progress = FALSE
)
```

---

as *as("sf", "ShapeMap")*

Description

This is a direct conversion, for ShapeMap -> Axial -> Segment see axialToSegmentShapeGraph

This is a direct conversion, for ShapeMap -> Axial -> Segment see axialToSegmentShapeGraph

See Also

Other ShapeMap: ShapeMap-class

Other ShapeMap: ShapeMap-class

Other AxialShapeGraph: AxialShapeGraph-class

Other AxialShapeGraph: AxialShapeGraph-class

Other SegmentShapeGraph: SegmentShapeGraph-class

Other SegmentShapeGraph: SegmentShapeGraph-class

axialAnalysisLocal *Axial analysis - local metrics*

Description

Runs axial analysis to get the local metrics Control and Controllability

Usage

```
axialAnalysisLocal(shapeGraph, copyMap = TRUE, verbose = FALSE)
```

Arguments

shapeGraph	An Axial ShapeGraph
copyMap	Optional. Copy the internal sala map
verbose	Optional. Show more information of the process.

Value

Returns a list with:

- completed: Whether the analysis completed
- newAttributes: The new attributes that were created during the process

Examples

```
mifFile <- system.file(
    "extdata", "testdata", "barnsbury",
    "barnsbury_small_axial_original.mif",
    package = "alcyon"
  )
  sfMap <- st_read(mifFile,
    geometry_column = 1L, quiet = TRUE
  )
  shapeGraph <- as(sfMap, "AxialShapeGraph")
axialAnalysisLocal(shapeGraph)
```

AxialShapeGraph-class *Axial ShapeGraph*

Description

A representation of sala's Axial ShapeGraph in R. Holds onto a sala Axial ShapeGraph pointer and operates on that

See Also

Other AxialShapeGraph: as()

AxialShapeGraph_subset

Subset AxialShapeGraph objects

Description

Subsetting AxialShapeGraph objects essentially passes the data to sf. See sf

Usage

```
## S3 method for class 'AxialShapeGraph'
x[...]

## S3 replacement method for class 'AxialShapeGraph'
x[...] <- value
```

Arguments

x	object of class AxialShapeGraph passed to stars[]
...	other parameters passed to stars[] <-
value	value to be passed to sf[] <-

axialToSegmentShapeGraph

Axial to Segment ShapeGraph

Description

Convert an Axial ShapeGraph to a Segment ShapeGraph

Usage

```
axialToSegmentShapeGraph(axialShapeGraph, stubRemoval = NULL)
```

Arguments

axialShapeGraph	An Axial ShapeGraph
stubRemoval	Remove stubs of axial lines shorter than this percentage (for example provide 0.4 for 40%)

Value

A new Segment ShapeGraph

Examples

```
mifFile <- system.file(
    "extdata", "testdata", "barnsbury",
    "barnsbury_small_axial_original.mif",
    package = "alcyon"
  )
  sfMap <- st_read(mifFile,
    geometry_column = 1L, quiet = TRUE
  )
  shapeGraph <- as(sfMap, "AxialShapeGraph")
axialToSegmentShapeGraph(shapeGraph, stubRemoval = 0.4)
```

`blockLines` *Block lines on a PointMap*

Description

Takes a PointMap and a ShapeMap with lines and blocks the cells on the PointMap where the lines pass.

Usage

```
blockLines(pointMap, lineStringMap, copyMap = TRUE, verbose = FALSE)
```

Arguments

`pointMap`	The input PointMap
`lineStringMap`	Map of lines, either a ShapeMap, or an sf lineString map
`copyMap`	Optional. Copy the internal sala map
`verbose`	Optional. Show more information of the process.

Value

A new PointMap with points as they have been blocked by the lines

Examples

```
mifFile <- system.file(
    "extdata", "testdata", "simple",
    "simple_interior.mif",
    package = "alcyon"
  )
  sfMap <- st_read(mifFile,
    geometry_column = 1L, quiet = TRUE
  )
  shapeMap <- as(sfMap[, vector()], "ShapeMap")
lineStringMap <- as(sfMap, "sf")
mapRegion <- sf::st_bbox(lineStringMap)
```

```
pointMap <- createGrid(
  minX = mapRegion[["xmin"]],
  minY = mapRegion[["ymin"]],
  maxX = mapRegion[["xmax"]],
  maxY = mapRegion[["ymax"]],
  gridSize = 0.04
)
blockLines(
  pointMap = pointMap,
  lineStringMap = lineStringMap[vector()]
)
```

`connections` *Get map connections*

Description

Get map connections

Usage

```
connections(map)
```

Arguments

`map`	A sala map

Value

A matrix with the connected refs

`connections,AxialShapeGraph-method` *Get the Axial ShapeGraph connections*

Description

Get the Axial ShapeGraph connections

Usage

```
## S4 method for signature 'AxialShapeGraph'
connections(map)
```

Arguments

`map`	An Axial ShapeGraph

Value

A matrix with the connected refs

Examples

```
mifFile <- system.file(
    "extdata", "testdata", "barnsbury",
    "barnsbury_small_axial_original.mif",
    package = "alcyon"
  )
  sfMap <- st_read(mifFile,
    geometry_column = 1L, quiet = TRUE
  )
  shapeGraph <- as(sfMap, "AxialShapeGraph")
connections(shapeGraph)
```

`connections,PointMap-method`
Get the PointMap connections

Description

Get the PointMap connections

Usage

```
## S4 method for signature 'PointMap'
connections(map)
```

Arguments

`map`	A PointMap

Value

A matrix with the connected refs

Examples

```
mifFile <- system.file(
    "extdata", "testdata", "gallery",
    "gallery_lines.mif",
    package = "alcyon"
  )
  sfMap <- st_read(mifFile,
    geometry_column = 1L, quiet = TRUE
  )
  pointMap <- makeVGAPointMap(
    sfMap,
```

```
    gridSize = 0.04,
    fillX = 3.01,
    fillY = 6.7,
    maxVisibility = NA,
    boundaryGraph = FALSE,
    verbose = FALSE
  )
# plot the first 100 connections only
head(connections(pointMap), 100)
```

---

connections,SegmentShapeGraph-method
Get the Segment ShapeGraph connections

Description

Get the Segment ShapeGraph connections

Usage

```
## S4 method for signature 'SegmentShapeGraph'
connections(map)
```

Arguments

map	An Segment ShapeGraph

Value

A matrix with the connected refs

Examples

```
mifFile <- system.file(
    "extdata", "testdata", "barnsbury",
    "barnsbury_small_segment_original.mif",
    package = "alcyon"
  )
  sfMap <- st_read(mifFile,
    geometry_column = 1L, quiet = TRUE
  )
  shapeGraph <- as(sfMap, "SegmentShapeGraph")
connections(shapeGraph)
```

createGrid *Create a PointMap through a grid*

Description

Create a PointMap through a grid

Usage

```
createGrid(minX, minY, maxX, maxY, gridSize, verbose = FALSE)
```

Arguments

minX	Minimum X of the bounding region
minY	Minimum Y of the bounding region
maxX	Maximum X of the bounding region
maxY	Maximum Y of the bounding region
gridSize	Size of the cells
verbose	Optional. Show more information of the process.

Value

A new PointMap

Examples

```
mifFile <- system.file(
    "extdata", "testdata", "simple",
    "simple_interior.mif",
    package = "alcyon"
  )
  sfMap <- st_read(mifFile,
    geometry_column = 1L, quiet = TRUE
  )
  shapeMap <- as(sfMap[, vector()], "ShapeMap")
lineStringMap <- as(sfMap, "sf")
mapRegion <- sf::st_bbox(lineStringMap)
createGrid(
  minX = mapRegion[["xmin"]],
  minY = mapRegion[["ymin"]],
  maxX = mapRegion[["xmax"]],
  maxY = mapRegion[["ymax"]],
  gridSize = 0.04
)
```

`fillGrid` *Fill a PointMap's grid starting from one or more points*

Description

Fill a PointMap's grid starting from one or more points

Usage

```
fillGrid(pointMap, fillX, fillY, copyMap = TRUE, verbose = FALSE)
```

Arguments

`pointMap`	The input PointMap
`fillX`	X coordinate of the fill points
`fillY`	Y coordinate of the fill points
`copyMap`	Optional. Copy the internal sala map
`verbose`	Optional. Show more information of the process.

Value

A new PointMap with filled points

Examples

```
mifFile <- system.file(
    "extdata", "testdata", "simple",
    "simple_interior.mif",
    package = "alcyon"
  )
  sfMap <- st_read(mifFile,
    geometry_column = 1L, quiet = TRUE
  )
  shapeMap <- as(sfMap[, vector()], "ShapeMap")
lineStringMap <- as(sfMap, "sf")
mapRegion <- sf::st_bbox(lineStringMap)
pointMap <- createGrid(
  minX = mapRegion[["xmin"]],
  minY = mapRegion[["ymin"]],
  maxX = mapRegion[["xmax"]],
  maxY = mapRegion[["ymax"]],
  gridSize = 0.04
)
pointMap <- blockLines(
  pointMap = pointMap,
  lineStringMap = lineStringMap[vector()]
)
fillGrid(
```

```
  pointMap = pointMap,
  fillX = 3.01,
  fillY = 6.7
)
```

`getTopFeatures` *Extract top x percent of features*

Description

Sorts features by a specific column and extracts the top x percent

Usage

```
getTopFeatures(lineStringMap, column, percent)
```

Arguments

`lineStringMap`	An sf lineString map
`column`	The column to use to extract the features from
`percent`	Percentage of features (to total) to extract

Value

The lineString map filtered and sorted

Examples

```
mifFile <- system.file(
    "extdata", "testdata", "barnsbury",
    "barnsbury_small_axial_original.mif",
    package = "alcyon"
  )
  sfMap <- st_read(mifFile,
    geometry_column = 1L, quiet = TRUE
  )
  shapeGraph <- as(sfMap, "AxialShapeGraph")
shapeGraph <- allToAllTraverse(
  shapeGraph,
  traversalType = TraversalType$Topological,
  radii = c("n", "3"),
  includeBetweenness = TRUE
)
getTopFeatures(shapeGraph, "Connectivity", 0.1)
```

`isovist` *Create isovists at point and direction angle*

Description

Create one or more isovists at particular points, given angle and field of view

Usage

```
isovist(boundaryMap, x, y, angle = NA, viewAngle = NA, verbose = FALSE)
```

Arguments

`boundaryMap`	A ShapeMap with lines designating the isovist boundaries
`x`	X coordinate of the origin points
`y`	Y coordinate of the origin points
`angle`	The angle (from the X axis) of the isovist look direction
`viewAngle`	The angle signifying the isovist's field of view
`verbose`	Optional. Show more information of the process.

Value

A ShapeMap with the isovist polygons

Examples

```
mifFile <- system.file(
    "extdata", "testdata", "simple",
    "simple_interior.mif",
    package = "alcyon"
  )
  sfMap <- st_read(mifFile,
    geometry_column = 1L, quiet = TRUE
  )
  shapeMap <- as(sfMap[, vector()], "ShapeMap")
isovist(
  shapeMap,
  x = c(3.01, 1.3),
  y = c(6.70, 5.2),
  angle = 0.01,
  viewAngle = 3.14,
  FALSE
)
```

isovist2pts *Create isovists using two points*

Description

Create one or more isovists at particular points, given another point for direction and an angle for field of view

Usage

```
isovist2pts(boundaryMap, x, y, toX, toY, viewAngle, verbose = FALSE)
```

Arguments

boundaryMap	A ShapeMap with lines designating the isovist boundaries
x	X coordinate of the origin points
y	Y coordinate of the origin points
toX	X coordinate of the target points
toY	Y coordinate of the target points
viewAngle	The angle signifying the isovist's field of view
verbose	Optional. Show more information of the process.

Value

A ShapeMap with the isovist polygons

Examples

```
mifFile <- system.file(
    "extdata", "testdata", "simple",
    "simple_interior.mif",
    package = "alcyon"
  )
  sfMap <- st_read(mifFile,
    geometry_column = 1L, quiet = TRUE
  )
  shapeMap <- as(sfMap[, vector()], "ShapeMap")
isovist2pts(
  shapeMap,
  x = c(3.01, 1.3),
  y = c(6.70, 5.2),
  toX = c(3.40, 1.1),
  toY = c(6.50, 5.6),
  viewAngle = 3.14,
  FALSE
)
```

`linkCoords` *Link map points/lines as if selecting them using points*

Description

Link map points/lines as if selecting them using points

Usage

```
linkCoords(map, fromX, fromY, toX, toY, copyMap = TRUE)
```

Arguments

`map`	A sala map
`fromX`	X coordinate of the origin point
`fromY`	Y coordinate of the origin point
`toX`	X coordinate of the target point
`toY`	Y coordinate of the target point
`copyMap`	Optional. Copy the internal sala map

Value

A new map with linked points/lines

`linkCoords,AxialShapeGraph-method`
Link two Axial Lines (coordinates)

Description

Link two locations on an Axial ShapeGraph using the point coordinates

Usage

```
## S4 method for signature 'AxialShapeGraph'
linkCoords(map, fromX, fromY, toX, toY, copyMap = TRUE)
```

Arguments

`map`	An Axial ShapeGraph
`fromX`	X coordinate of the first link point
`fromY`	Y coordinate of the first link point
`toX`	X coordinate of the second link point
`toY`	Y coordinate of the second link point
`copyMap`	Optional. Copy the internal sala map

Value

A new Axial ShapeGraph with linked lines

Examples

```
mifFile <- system.file(
    "extdata", "testdata", "barnsbury",
    "barnsbury_small_axial_original.mif",
    package = "alcyon"
  )
  sfMap <- st_read(mifFile,
    geometry_column = 1L, quiet = TRUE
  )
  shapeGraph <- as(sfMap, "AxialShapeGraph")
linkCoords(shapeGraph, 982.8, -1620.3, 1217.1, -1977.3)
```

---

linkCoords,PointMap-method

Link two PointMap Cells (coordinates)

Description

Link two cells on a PointMap using the point coordinates

Usage

```
## S4 method for signature 'PointMap'
linkCoords(map, fromX, fromY, toX, toY, copyMap = TRUE)
```

Arguments

map	A PointMap
fromX	X coordinate of the first link point
fromY	Y coordinate of the first link point
toX	X coordinate of the second link point
toY	Y coordinate of the second link point
copyMap	Optional. Copy the internal sala map

Value

A new PointMap with linked points

Examples

```
mifFile <- system.file(
    "extdata", "testdata", "gallery",
    "gallery_lines.mif",
    package = "alcyon"
  )
  sfMap <- st_read(mifFile,
    geometry_column = 1L, quiet = TRUE
  )
  pointMap <- makeVGAPointMap(
    sfMap,
    gridSize = 0.04,
    fillX = 3.01,
    fillY = 6.7,
    maxVisibility = NA,
    boundaryGraph = FALSE,
    verbose = FALSE
  )
linkCoords(pointMap, 1.74, 6.7, 5.05, 5.24)
```

---

linkRefs *Link map points/lines using their refs*

Description

Link map points/lines using their refs

Usage

```
linkRefs(map, fromRef, toRef, copyMap = TRUE)
```

Arguments

map	A sala map
fromRef	The ref of the origin element
toRef	The ref of the target element
copyMap	Optional. Copy the internal sala map

Value

A new map with linked points/lines

linkRefs,AxialShapeGraph-method
Link two Axial Lines (refs)

Description

Link two lines on an Axial ShapeGraph using their refs

Usage

```
## S4 method for signature 'AxialShapeGraph'
linkRefs(map, fromRef, toRef, copyMap = TRUE)
```

Arguments

map	An Axial ShapeGraph
fromRef	Ref of the first link line
toRef	Ref of the second link line
copyMap	Optional. Copy the internal sala map

Value

A new Axial ShapeGraph with linked lines

Examples

```
mifFile <- system.file(
    "extdata", "testdata", "barnsbury",
    "barnsbury_small_axial_original.mif",
    package = "alcyon"
  )
  sfMap <- st_read(mifFile,
    geometry_column = 1L, quiet = TRUE
  )
  shapeGraph <- as(sfMap, "AxialShapeGraph")
linkRefs(shapeGraph, 0L, 9L)
```

linkRefs,PointMap-method
Link two PointMap Cells (refs)

Description

Link two cells on an PointMap using their refs

Usage

```
## S4 method for signature 'PointMap'
linkRefs(map, fromRef, toRef, copyMap = TRUE)
```

Arguments

map	A PointMap
fromRef	Ref of the first link line
toRef	Ref of the second link line
copyMap	Optional. Copy the internal sala map

Value

A new PointMap with linked points

Examples

```
mifFile <- system.file(
    "extdata", "testdata", "gallery",
    "gallery_lines.mif",
    package = "alcyon"
  )
  sfMap <- st_read(mifFile,
    geometry_column = 1L, quiet = TRUE
  )
  pointMap <- makeVGAPointMap(
    sfMap,
    gridSize = 0.04,
    fillX = 3.01,
    fillY = 6.7,
    maxVisibility = NA,
    boundaryGraph = FALSE,
    verbose = FALSE
  )
pointMap <- linkRefs(pointMap, 1835056L, 7208971L)
```

---

links *Get map links*

Description

Get map links

Usage

```
links(map)
```

Arguments

map	A sala map

Value

A matrix with the linked refs

`links,AxialShapeGraph-method`
Get the Axial ShapeGraph links

Description

Get the Axial ShapeGraph links

Usage

```
## S4 method for signature 'AxialShapeGraph'
links(map)
```

Arguments

map	An Axial ShapeGraph

Value

A matrix with the linked refs

Examples

```
# links of an axial map
mifFile <- system.file(
    "extdata", "testdata", "barnsbury",
    "barnsbury_small_axial_original.mif",
    package = "alcyon"
  )
  sfMap <- st_read(mifFile,
    geometry_column = 1L, quiet = TRUE
  )
  shapeGraph <- as(sfMap, "AxialShapeGraph")
linkRefs(shapeGraph, 0L, 9L)
unlinkCoords(shapeGraph, 530923.0, 184041.0, 530956.0, 183887.0)
links(shapeGraph)
```

links,PointMap-method *Get the PointMap links*

Description

Get the PointMap links

Usage

```
## S4 method for signature 'PointMap'
links(map)
```

Arguments

map	A PointMap

Value

A matrix with the linked refs

Examples

```
mifFile <- system.file(
    "extdata", "testdata", "gallery",
    "gallery_lines.mif",
    package = "alcyon"
  )
  sfMap <- st_read(mifFile,
    geometry_column = 1L, quiet = TRUE
  )
  pointMap <- makeVGAPointMap(
    sfMap,
    gridSize = 0.04,
    fillX = 3.01,
    fillY = 6.7,
    maxVisibility = NA,
    boundaryGraph = FALSE,
    verbose = FALSE
  )
linkRefs(pointMap, 1835056L, 7208971L)
links(pointMap)
```

makeAllLineMap *Create an All-line Map*

Description

Create an All-line Map

Usage

```
makeAllLineMap(boundsMap, seedX, seedY, verbose = FALSE)
```

Arguments

boundsMap	The boundary ShapeMap to create the all-line map in
seedX	X coordinate of the seed (the point that initiates the process)
seedY	Y coordinate of the seed (the point that initiates the process)
verbose	Optional. Show more information of the process.

Value

An All-line Axial ShapeGraph

Examples

```
mifFile <- system.file(
    "extdata", "testdata", "simple",
    "simple_interior.mif",
    package = "alcyon"
  )
  sfMap <- st_read(mifFile,
    geometry_column = 1L, quiet = TRUE
  )
  shapeMap <- as(sfMap[, vector()], "ShapeMap")
makeAllLineMap(
  shapeMap,
  seedX = 3.01,
  seedY = 6.7
)
```

makeColour *Single Colour from depthmapX's Palettes*

Description

Create a single colour from depthmapX's palettes.

Usage

```
makeDepthmapClassicColour(value, rangeMin = 0, rangeMax = 1)

makeAxmanesqueColour(value, rangeMin = 0, rangeMax = 1)

makePurpleOrangeColour(value, rangeMin = 0, rangeMax = 1)

makeBlueRedColour(value, rangeMin = 0, rangeMax = 1)

makeGreyScaleColour(value, rangeMin = 0, rangeMax = 1)

makeNiceHSBColour(value, rangeMin = 0, rangeMax = 1)
```

Arguments

value	Value within the min/max range to take
rangeMin	The min value of the range
rangeMax	The max value of the range

Value

Returns a single colour.

Examples

```
makeDepthmapClassicColour(0.2, 0, 1)
makeAxmanesqueColour(0.2, 0, 1)
makePurpleOrangeColour(0.2, 0, 1)
makeBlueRedColour(0.2, 0, 1)
makeGreyScaleColour(0.2, 0, 1)
makeNiceHSBColour(0.2, 0, 1)
```

makeVGAGraph *Create a graph between visible cells in the PointMap*

Description

Create a graph between visible cells in the PointMap

Usage

```
makeVGAGraph(
  pointMap,
  boundaryGraph = FALSE,
  maxVisibility = NA,
  copyMap = TRUE,
  verbose = FALSE
)
```

Arguments

pointMap	The input PointMap
boundaryGraph	Only create a graph on the boundary cells
maxVisibility	Limit how far two cells can be to be connected
copyMap	Optional. Copy the internal sala map
verbose	Optional. Show more information of the process.

Value

A new PointMap with a graph between points

Examples

```
mifFile <- system.file(
    "extdata", "testdata", "simple",
    "simple_interior.mif",
    package = "alcyon"
  )
  sfMap <- st_read(mifFile,
    geometry_column = 1L, quiet = TRUE
  )
  shapeMap <- as(sfMap[, vector()], "ShapeMap")
lineStringMap <- as(sfMap, "sf")
mapRegion <- sf::st_bbox(lineStringMap)
pointMap <- createGrid(
  minX = mapRegion[["xmin"]],
  minY = mapRegion[["ymin"]],
  maxX = mapRegion[["xmax"]],
  maxY = mapRegion[["ymax"]],
```

```
  gridSize = 0.5
)
pointMap <- blockLines(
  pointMap = pointMap,
  lineStringMap = lineStringMap[vector()]
)
pointMap <- fillGrid(
  pointMap = pointMap,
  fillX = 3.01,
  fillY = 6.7
)
makeVGAGraph(
  pointMap = pointMap,
  boundaryGraph = FALSE,
  maxVisibility = NA
)
```

`makeVGAPointMap` *Create a PointMap grid, fill it and make the graph*

Description

This is intended to be a single command to get from the lines to a PointMap ready for analysis

Usage

```
makeVGAPointMap(
  lineStringMap,
  gridSize,
  fillX,
  fillY,
  maxVisibility = NA,
  boundaryGraph = FALSE,
  verbose = FALSE
)
```

Arguments

`lineStringMap`	Map of lines, either a ShapeMap, or an sf lineString map
`gridSize`	Size of the cells
`fillX`	X coordinate of the fill points
`fillY`	Y coordinate of the fill points
`maxVisibility`	Limit how far two cells can be to be connected
`boundaryGraph`	Only create a graph on the boundary cells
`verbose`	Optional. Show more information of the process.

Value

A new PointMap

Examples

```
mifFile <- system.file(
    "extdata", "testdata", "simple",
    "simple_interior.mif",
    package = "alcyon"
  )
  sfMap <- st_read(mifFile,
    geometry_column = 1L, quiet = TRUE
  )
  shapeMap <- as(sfMap[, vector()], "ShapeMap")
makeVGAPointMap(
  sfMap,
  gridSize = 0.5,
  fillX = 3.01,
  fillY = 6.7,
  maxVisibility = NA,
  boundaryGraph = FALSE,
  verbose = FALSE
)
```

---

matchPointsToLines *Match points to lines*

Description

Match points to their closest line. Matches (spatial-join) points to lines. Finds the point closest to a line. One point is attached to one line, thus if fewer points than lines are given then some lines will have no point attached.

Usage

```
matchPointsToLines(points, lines, getIndex = FALSE)
```

Arguments

points	Points to attach.
lines	Lines to attach to.
getIndex	Get the index returned and not the data.

Value

If getIndex is TRUE then the index of the points as they relate to the matching lines are given. If not, then the data from the points dataframe is returned.

Examples

```
segmentsMif <- system.file(
    "extdata", "testdata", "barnsbury",
    "barnsbury_small_segment_original.mif",
    package = "alcyon"
)
segmentsSf <- st_read(
    segmentsMif,
    geometry_column = 1L, quiet = TRUE
)
gateCountsMif <- system.file(
    "extdata", "testdata", "barnsbury",
    "barnsbury_ped_gatecounts.mif",
    package = "alcyon"
)
gateCountsSf <- st_read(
    gateCountsMif,
    geometry_column = 1L, quiet = TRUE
)
matchPointsToLines(gateCountsSf, segmentsSf)
```

---

name *Get map name*

Description

Get map name

Usage

```
name(map)
```

Arguments

map	A sala map

Value

The name of the object as a string

name,PointMap-method *Get the PointMap name*

Description

Get the PointMap name

Usage

```
## S4 method for signature 'PointMap'
name(map)
```

Arguments

map	A PointMap

Value

The name of the PointMap as a string

Examples

```
mifFile <- system.file(
    "extdata", "testdata", "gallery",
    "gallery_lines.mif",
    package = "alcyon"
  )
  sfMap <- st_read(mifFile,
    geometry_column = 1L, quiet = TRUE
  )
  pointMap <- makeVGAPointMap(
    sfMap,
    gridSize = 0.04,
    fillX = 3.01,
    fillY = 6.7,
    maxVisibility = NA,
    boundaryGraph = FALSE,
    verbose = FALSE
  )
name(pointMap)
```

name,ShapeMap-method *Get the ShapeMap name*

Description

Get the ShapeMap name

Usage

```
## S4 method for signature 'ShapeMap'
name(map)
```

Arguments

map	A ShapeMap

Value

The name of the ShapeMap as a string

Examples

```
mifFile <- system.file(
    "extdata", "testdata", "simple",
    "simple_interior.mif",
    package = "alcyon"
  )
  sfMap <- st_read(mifFile,
    geometry_column = 1L, quiet = TRUE
  )
  shapeMap <- as(sfMap[, vector()], "ShapeMap")
name(shapeMap)
```

oneToAllTraverse *One-to-all traversal*

Description

Runs one-to-all traversal on a map with a graph. This is applicable to:

- PointMaps (Visibility Graph Analysis)
- Axial ShapeGraphs (Axial analysis)
- Segment ShapeGraphs (Segment analysis)

Usage

```
oneToAllTraverse(
  map,
  traversalType,
  fromX,
  fromY,
  quantizationWidth = NA,
  copyMap = TRUE,
  verbose = FALSE
)
```

Arguments

map	A PointMap, Axial ShapeGraph or Segment ShapeGraph
traversalType	The traversal type. See TraversalType
fromX	X coordinate of the point to start the traversal from
fromY	X coordinate of the point to start the traversal from
quantizationWidth	Set this to use chunks of this width instead of continuous values for the cost of traversal. This is equivalent to the "tulip bins" for depthmapX's tulip analysis (1024 tulip bins = pi/1024 quantizationWidth). Only works for Segment ShapeGraphs
copyMap	Optional. Copy the internal sala map
verbose	Optional. Show more information of the process.

Value

Returns a list with:

- completed: Whether the analysis completed
- newAttributes: The new attributes that were created during the process

Examples

```
# Pointmap analysis (VGA)
mifFile <- system.file(
    "extdata", "testdata", "simple",
    "simple_interior.mif",
    package = "alcyon"
  )
  sfMap <- st_read(mifFile,
    geometry_column = 1L, quiet = TRUE
  )
  pointMap <- makeVGAPointMap(
    sfMap,
    gridSize = 0.5,
    fillX = 3.0,
    fillY = 6.0,
```

```
    maxVisibility = NA,
    boundaryGraph = FALSE,
    verbose = FALSE
  )
oneToAllTraverse(
  pointMap,
  traversalType = TraversalType$Metric,
  fromX = 3.01,
  fromY = 6.7
)

# Axial analysis
mifFile <- system.file(
    "extdata", "testdata", "barnsbury",
    "barnsbury_small_axial_original.mif",
    package = "alcyon"
  )
  sfMap <- st_read(mifFile,
    geometry_column = 1L, quiet = TRUE
  )
  shapeGraph <- as(sfMap, "AxialShapeGraph")
oneToAllTraverse(
  shapeGraph,
  traversalType = TraversalType$Topological,
  fromX = 1217.1,
  fromY = -1977.3
)

# Segment analysis
mifFile <- system.file(
    "extdata", "testdata", "barnsbury",
    "barnsbury_small_segment_original.mif",
    package = "alcyon"
  )
  sfMap <- st_read(mifFile,
    geometry_column = 1L, quiet = TRUE
  )
  shapeGraph <- as(sfMap, "SegmentShapeGraph")
oneToAllTraverse(
  shapeGraph,
  traversalType = TraversalType$Topological,
  fromX = 1217.1,
  fromY = -1977.3
)
```

`oneToOneTraverse` *One-to-one traversal*

Description

Runs one-to-one traversal on a map with a graph. This is applicable to:

- PointMaps (Visibility Graph Analysis)
- Segment ShapeGraphs (Segment analysis)

Usage

```
oneToOneTraverse(
  map,
  traversalType,
  fromX,
  fromY,
  toX,
  toY,
  quantizationWidth = NA,
  copyMap = TRUE,
  verbose = FALSE
)
```

Arguments

map	A PointMap or Segment ShapeGraph
traversalType	The traversal type. See TraversalType
fromX	X coordinate of the point(s) to start the traversal from
fromY	X coordinate of the point(s) to start the traversal from
toX	X coordinate of the point(s) to start the traversal from
toY	X coordinate of the point(s) to start the traversal from
quantizationWidth	Set this to use chunks of this width instead of continuous values for the cost of traversal. This is equivalent to the "tulip bins" for depthmapX's tulip analysis (1024 tulip bins = pi/1024 quantizationWidth). Only works for Segment ShapeGraphs
copyMap	Optional. Copy the internal sala map
verbose	Optional. Show more information of the process.

Value

Returns a list with:

- completed: Whether the analysis completed
- newAttributes: The new attributes that were created during the process

Examples

```
# Pointmap analysis (VGA)
mifFile <- system.file(
    "extdata", "testdata", "simple",
    "simple_interior.mif",
    package = "alcyon"
```

```
  )
  sfMap <- st_read(mifFile,
    geometry_column = 1L, quiet = TRUE
  )
  pointMap <- makeVGAPointMap(
    sfMap,
    gridSize = 0.5,
    fillX = 3.0,
    fillY = 6.0,
    maxVisibility = NA,
    boundaryGraph = FALSE,
    verbose = FALSE
  )
oneToOneTraverse(
  pointMap,
  traversalType = TraversalType$Metric,
  fromX = 7.52,
  fromY = 6.02,
  toX = 5.78,
  toY = 2.96
)

# Segment analysis
mifFile <- system.file(
    "extdata", "testdata", "barnsbury",
    "barnsbury_small_segment_original.mif",
    package = "alcyon"
  )
  sfMap <- st_read(mifFile,
    geometry_column = 1L, quiet = TRUE
  )
  shapeGraph <- as(sfMap, "SegmentShapeGraph")
oneToOneTraverse(
  shapeGraph,
  traversalType = TraversalType$Topological,
  fromX = 1217.1,
  fromY = -1977.3,
  toX = 1017.8,
  toY = -1699.3
)
```

---

palettes *Colour Palettes from depthmapX*

Description

Create n contiguous colours taken from depthmapX.

Usage

```
depthmap.classic.colour(n, rangeMin = 0, rangeMax = 1)

depthmap.axmanesque.colour(n, rangeMin = 0, rangeMax = 1)

depthmap.purpleorange.colour(n, rangeMin = 0, rangeMax = 1)

depthmap.bluered.colour(n, rangeMin = 0, rangeMax = 1)

depthmap.grayscale.colour(n, rangeMin = 0, rangeMax = 1)

depthmap.nicehsb.colour(n, rangeMin = 0, rangeMax = 1)
```

Arguments

n	Number of colours to generate
rangeMin	The min value of the range
rangeMax	The max value of the range

Value

Returns a vector of colours.

Examples

```
depthmap.classic.colour(100, 0, 1)
depthmap.axmanesque.colour(100, 0, 1)
depthmap.purpleorange.colour(100, 0, 1)
depthmap.bluered.colour(100, 0, 1)
depthmap.grayscale.colour(100, 0, 1)
depthmap.nicehsb.colour(100, 0, 1)
```

---

plot.PointMap *plot a PointMap*

Description

Calls a standard plot.stars, but flips the first argument around the x axis

Usage

```
## S3 method for class 'PointMap'
plot(x, ...)
```

Arguments

x	object of class PointMap
...	other parameters passed to stars[]

PointMap-class *PointMap*

Description

A representation of sala's PointMap in R. Holds onto a sala PointMap pointer and operates on that

PointMap_subset *Subset PointMap objects*

Description

Subsetting PointMap objects essentially passes the data to stars See stars_subset

Usage

```
## S3 method for class 'PointMap'
x[...]

## S3 replacement method for class 'PointMap'
x[...] <- value
```

Arguments

x	object of class PointMap passed to stars[]
...	other parameters passed to stars[] <-
value	value to be passed to stars[] <-

readMetaGraph *Read MetaGraph*

Description

Reads a metagraph into a bunch of ShapeMaps/ShapeGraphs/PointMaps

Usage

```
readMetaGraph(fileName)
```

Arguments

fileName	The metagraph file

Value

A list of ShapeMaps, ShapeGraphs and PointMaps

Examples

```
fileName <- system.file(
    "extdata", "testdata", "barnsbury", "barnsburySmall.graph",
    package = "alcyon"
)
readMetaGraph(fileName)
```

`reduceToFewest` *Reduce an All-line Map to two types of fewest-line maps*

Description

Reduce an All-line Map to two types of fewest-line maps

Usage

```
reduceToFewest(allLineMap)
```

Arguments

`allLineMap`	An AllLineShapeGraph

Value

A list with two fewest-line axial ShapeGraphs

Examples

```
mifFile <- system.file(
    "extdata", "testdata", "simple",
    "simple_interior.mif",
    package = "alcyon"
  )
  sfMap <- st_read(mifFile,
    geometry_column = 1L, quiet = TRUE
  )
  shapeMap <- as(sfMap[, vector()], "ShapeMap")
allLineMap <- makeAllLineMap(
  shapeMap,
  seedX = 3.01,
  seedY = 6.7
)
reduceToFewest(allLineMap)
```

refIdToIndexAndBack *Ref ID to index and vice-versa*

Description

Converts a depthmapX "Ref" ID to the indices (x, y) of a cell, or the reverse

Usage

```
refIDtoIndex(refID)

indexToRefID(i, j)
```

Arguments

refID	The Ref ID
i	The x-axis index of the cell
j	The y-axis index of the cell

Value

A pair of indices (x, y) or a Ref ID

Examples

```
idx <- refIDtoIndex(852645)
# outputs:
#    i   j
# 1 13 677

idx <- indexToRefID(13, 667)
# outputs:
# 852645
```

SegmentShapeGraph-class *Segment ShapeGraph*

Description

A representation of sala's Segment ShapeGraph in R. Holds onto a sala Segment ShapeGraph pointer and operates on that

See Also

Other SegmentShapeGraph: as()

SegmentShapeGraph_subset *Subset SegmentShapeGraph objects*

Description

Subsetting SegmentShapeGraph objects essentially passes the data to sf. See sf

Usage

```
## S3 method for class 'SegmentShapeGraph'
x[...]

## S3 replacement method for class 'SegmentShapeGraph'
x[...] <- value
```

Arguments

x	object of class SegmentShapeGraph passed to stars[]
...	other parameters passed to stars[] <-
value	value to be passed to sf[] <-

shapegraphToGraphData *Conversion of shapegraph to graph data*

Description

Creates data to be construct a graph, based on the connections and the x,y coordinates of the centroids of shapes in a shapegraph (axial, segment, convex). Specify weightColumn to assign weight to graph edges.

Usage

```
shapegraphToGraphData(shapeGraph, weightColumn = NA)
```

Arguments

shapeGraph	A ShapeGraph
weightColumn	Optional. The variable used to assign weight to graph edges

Details

If weightColumn is provided, edge connections weight is calculated by taking the average of the variable of the connected nodes.

Value

Returns a list with edges and vertices for constructing a graph.

Examples

```
mifFile <- system.file(
    "extdata", "testdata", "barnsbury",
    "barnsbury_small_axial_original.mif",
    package = "alcyon"
  )
  sfMap <- st_read(mifFile,
    geometry_column = 1L, quiet = TRUE
  )
  shapeGraph <- as(sfMap, "AxialShapeGraph")
shapegraphToGraphData(shapeGraph)
```

ShapeMap-class *ShapeMap class*

Description

A representation of sala's ShapeMap in R. Holds onto a sala ShapeMap pointer and operates on that

See Also

Other ShapeMap: `as()`

---

shapeMapToPolygonSf *ShapeMap to sf Polygon map*

Description

Convert a ShapeMap to an sf Polygon map

Usage

```
shapeMapToPolygonSf(shapeMap)
```

Arguments

shapeMap	A ShapeMap

Value

A new sf Polygon map

Examples

```
mifFile <- system.file(
    "extdata", "testdata", "simple",
    "simple_interior.mif",
    package = "alcyon"
  )
  sfMap <- st_read(mifFile,
    geometry_column = 1L, quiet = TRUE
  )
  shapeMap <- as(sfMap[, vector()], "ShapeMap")
isovistMap <- isovist(
  shapeMap,
  x = c(3.01, 1.3),
  y = c(6.70, 5.2),
  angle = 0.01,
  viewAngle = 3.14,
  FALSE
)
shapeMapToPolygonSf(isovistMap)
```

ShapeMap_subset *Subset ShapeMap objects*

Description

Subsetting ShapeMap objects essentially passes the data to sf. See sf

Usage

```
## S3 method for class 'ShapeMap'
x[...]

## S3 replacement method for class 'ShapeMap'
x[...] <- value
```

Arguments

x	object of class ShapeMap passed to sf[]
...	other parameters passed to sf[] <-
value	value to be passed to sf[] <-

TraversalType *Traversal types*

Description

These are meant to be used to indicate what kind of analysis is meant to be carried out.

Usage

```
TraversalType
```

Format

An object of class `list` of length 4.

Value

A list of numbers representing each particular analysis type

Examples

```
TraversalType$Angular
TraversalType$Topological
TraversalType$Metric
```

unlinkAtCrossPoint *Unlink map lines at their crossing point*

Description

Unlink map lines at their crossing point

Usage

```
unlinkAtCrossPoint(map, x, y, copyMap = TRUE)
```

Arguments

`map`	A sala map
`x`	X coordinate of the crossing point
`y`	Y coordinate of the crossing point
`copyMap`	Optional. Copy the internal sala map

Value

A new map with linked lines

unlinkAtCrossPoint,AxialShapeGraph-method *Unlink two Axial Lines (crosspoint)*

Description

Unlink two crossing lines on an Axial ShapeGraph at the crossing point

Usage

```
## S4 method for signature 'AxialShapeGraph'
unlinkAtCrossPoint(map, x, y, copyMap = TRUE)
```

Arguments

map	An Axial ShapeGraph
x	X coordinate of the unlink crossing point
y	Y coordinate of the unlink crossing point
copyMap	Optional. Copy the internal sala map

Value

A new Axial ShapeGraph with unlinked lines

Examples

```
mifFile <- system.file(
    "extdata", "testdata", "barnsbury",
    "barnsbury_small_axial_original.mif",
    package = "alcyon"
  )
  sfMap <- st_read(mifFile,
    geometry_column = 1L, quiet = TRUE
  )
  shapeGraph <- as(sfMap, "AxialShapeGraph")
unlinkAtCrossPoint(shapeGraph, 530925.0, 184119.0)
```

unlinkCoords *Unlink map points/lines as if selecting them using points*

Description

Unlink map points/lines as if selecting them using points

Usage

```
unlinkCoords(map, fromX, fromY, toX, toY, copyMap = TRUE)
```

Arguments

map	A sala map
fromX	X coordinate of the origin point
fromY	Y coordinate of the origin point
toX	X coordinate of the target point
toY	Y coordinate of the target point
copyMap	Optional. Copy the internal sala map

Value

A new map with unlinked points/lines

---

unlinkCoords,AxialShapeGraph-method

Unlink two Axial Lines (coordinates)

Description

Unlink two locations on an Axial ShapeGraph using the point coordinates

Usage

```
## S4 method for signature 'AxialShapeGraph'
unlinkCoords(map, fromX, fromY, toX, toY, copyMap = TRUE)
```

Arguments

map	An Axial ShapeGraph
fromX	X coordinate of the first unlink point
fromY	Y coordinate of the first unlink point
toX	X coordinate of the second unlink point
toY	Y coordinate of the second unlink point
copyMap	Optional. Copy the internal sala map

Value

A new Axial ShapeGraph with unlinked lines

Examples

```
mifFile <- system.file(
    "extdata", "testdata", "barnsbury",
    "barnsbury_small_axial_original.mif",
    package = "alcyon"
  )
  sfMap <- st_read(mifFile,
    geometry_column = 1L, quiet = TRUE
  )
  shapeGraph <- as(sfMap, "AxialShapeGraph")
unlinkCoords(shapeGraph, 982.8, -1620.3, 1080.4, -1873.5)
```

---

unlinkCoords,PointMap-method

Unlink two PointMap Cells (coordinates)

Description

Unlink two cells on a PointMap using the point coordinates

Usage

```
## S4 method for signature 'PointMap'
unlinkCoords(map, fromX, fromY, toX, toY, copyMap = TRUE)
```

Arguments

map	A PointMap
fromX	X coordinate of the first unlink point
fromY	Y coordinate of the first unlink point
toX	X coordinate of the second unlink point
toY	Y coordinate of the second unlink point
copyMap	Optional. Copy the internal sala map

Value

A new PointMap with unlinked points

Examples

```
mifFile <- system.file(
    "extdata", "testdata", "gallery",
    "gallery_lines.mif",
    package = "alcyon"
  )
  sfMap <- st_read(mifFile,
    geometry_column = 1L, quiet = TRUE
```

```
  )
  pointMap <- makeVGAPointMap(
    sfMap,
    gridSize = 0.04,
    fillX = 3.01,
    fillY = 6.7,
    maxVisibility = NA,
    boundaryGraph = FALSE,
    verbose = FALSE
  )
pointMap <- linkCoords(pointMap, 1.74, 6.7, 5.05, 5.24)
pointMap <- unlinkCoords(pointMap, 1.74, 6.7, 5.05, 5.24)
```

`unlinkRefs` *Unlink map points/lines using their refs*

Description

Unlink map points/lines using their refs

Usage

```
unlinkRefs(map, fromRef, toRef, copyMap = TRUE)
```

Arguments

`map`	A sala map
`fromRef`	The ref of the origin element
`toRef`	The ref of the target element
`copyMap`	Optional. Copy the internal sala map

Value

A new map with unlinked points/lines

`unlinkRefs,AxialShapeGraph-method` *Unlink two Axial Lines (refs)*

Description

Unlink two lines on an Axial ShapeGraph using their refs

Usage

```
## S4 method for signature 'AxialShapeGraph'
unlinkRefs(map, fromRef, toRef, copyMap = TRUE)
```

Arguments

map	An Axial ShapeGraph
fromRef	Ref of the first unlink line
toRef	Ref of the second unlink line
copyMap	Optional. Copy the internal sala map

Value

A new Axial ShapeGraph with unlinked lines

Examples

```
mifFile <- system.file(
    "extdata", "testdata", "barnsbury",
    "barnsbury_small_axial_original.mif",
    package = "alcyon"
  )
  sfMap <- st_read(mifFile,
    geometry_column = 1L, quiet = TRUE
  )
  shapeGraph <- as(sfMap, "AxialShapeGraph")
unlinkRefs(shapeGraph, 12L, 34L)
```

---

unlinkRefs,PointMap-method

Unlink two PointMap Cells (refs)

Description

Unlink two cells on an PointMap using their refs

Usage

```
## S4 method for signature 'PointMap'
unlinkRefs(map, fromRef, toRef, copyMap = TRUE)
```

Arguments

map	A PointMap
fromRef	Ref of the first unlink line
toRef	Ref of the second unlink line
copyMap	Optional. Copy the internal sala map

Value

A new PointMap with unlinked points

Examples

```
mifFile <- system.file(
    "extdata", "testdata", "gallery",
    "gallery_lines.mif",
    package = "alcyon"
  )
  sfMap <- st_read(mifFile,
    geometry_column = 1L, quiet = TRUE
  )
  pointMap <- makeVGAPointMap(
    sfMap,
    gridSize = 0.04,
    fillX = 3.01,
    fillY = 6.7,
    maxVisibility = NA,
    boundaryGraph = FALSE,
    verbose = FALSE
  )
pointMap <- linkRefs(pointMap, 1835056L, 7208971L)
pointMap <- unlinkRefs(pointMap, 1835056L, 7208971L)
```

---

unmakeVGAGraph *Unmake the graph in a PointMap*

Description

Unmake the graph in a PointMap

Usage

```
unmakeVGAGraph(pointMap, removeLinks = FALSE, copyMap = TRUE, verbose = FALSE)
```

Arguments

pointMap	The input PointMap
removeLinks	Also remove the links
copyMap	Optional. Copy the internal sala map
verbose	Optional. Show more information of the process.

Value

A new PointMap without the points graph

Examples

```
mifFile <- system.file(
    "extdata", "testdata", "simple",
    "simple_interior.mif",
    package = "alcyon"
  )
  sfMap <- st_read(mifFile,
    geometry_column = 1L, quiet = TRUE
  )
  shapeMap <- as(sfMap[, vector()], "ShapeMap")
pointMap <- makeVGAPointMap(
  sfMap,
  gridSize = 0.5,
  fillX = 3.01,
  fillY = 6.7,
  maxVisibility = NA,
  boundaryGraph = FALSE,
  verbose = FALSE
)
unmakeVGAGraph(
  pointMap = pointMap,
  removeLinks = FALSE
)
```

`vgaIsovist` *Visibility Graph Analysis - isovist metrics*

Description

Runs axial analysis to get the local metrics Control and Controllability

Usage

```
vgaIsovist(pointMap, boundaryMap, copyMap = TRUE)
```

Arguments

`pointMap`	A PointMap
`boundaryMap`	A ShapeMap of lines
`copyMap`	Optional. Copy the internal sala map

Value

A new PointMap with the results included

Examples

```
mifFile <- system.file(
    "extdata", "testdata", "simple",
    "simple_interior.mif",
    package = "alcyon"
  )
  sfMap <- st_read(mifFile,
    geometry_column = 1L, quiet = TRUE
  )
  pointMap <- makeVGAPointMap(
    sfMap,
    gridSize = 0.5,
    fillX = 3.0,
    fillY = 6.0,
    maxVisibility = NA,
    boundaryGraph = FALSE,
    verbose = FALSE
  )
boundaryMap <- as(sfMap[, c()], "ShapeMap")
vgaIsovist(pointMap, boundaryMap)
```

VGALocalAlgorithm *VGA Local Analysis algorithms.*

Description

Different algorithms for calculating the VGA Local metrics (Control, Controllability, Clustering Coefficient).

- VGALocalAlgorithm$None
- VGALocalAlgorithm$Standard
- VGALocalAlgorithm$AdjacencyMatrix

Usage

```
VGALocalAlgorithm
```

Format

An object of class `list` of length 3.

Value

A list of numbers representing each algorithm

Examples

```
VGALocalAlgorithm$Angular
VGALocalAlgorithm$Topological
VGALocalAlgorithm$Metric
```

vgaThroughVision *Visibility Graph Analysis - Through Vision*

Description

Runs Visibility Graph Analysis to get the Through Vision metric

Usage

```
vgaThroughVision(pointMap, copyMap = TRUE)
```

Arguments

pointMap	A PointMap
copyMap	Optional. Copy the internal sala map

Value

A new PointMap with the results included

Examples

```
mifFile <- system.file(
    "extdata", "testdata", "simple",
    "simple_interior.mif",
    package = "alcyon"
  )
  sfMap <- st_read(mifFile,
    geometry_column = 1L, quiet = TRUE
  )
  pointMap <- makeVGAPointMap(
    sfMap,
    gridSize = 0.5,
    fillX = 3.0,
    fillY = 6.0,
    maxVisibility = NA,
    boundaryGraph = FALSE,
    verbose = FALSE
  )
vgaThroughVision(pointMap)
```

vgaVisualLocal *Visibility Graph Analysis - Visual local metrics*

Description

Runs Visibility Graph Analysis to get visual local metrics

Usage

```
vgaVisualLocal(
  pointMap,
  nthreads = 1L,
  algorithm = VGALocalAlgorithm$Standard,
  copyMap = TRUE,
  gatesOnly = FALSE,
  progress = FALSE
)
```

Arguments

pointMap	A PointMap
nthreads	Optional. Number of threads to use (defaults to 1)
algorithm	Optional. The algorithm to use. See ?VGALocalAlgorithm
copyMap	Optional. Copy the internal sala map
gatesOnly	Optional. Only keep the values at specific gates
progress	Optional. Enable progress display

Value

A new PointMap with the results included

Examples

```
mifFile <- system.file(
    "extdata", "testdata", "simple",
    "simple_interior.mif",
    package = "alcyon"
  )
  sfMap <- st_read(mifFile,
    geometry_column = 1L, quiet = TRUE
  )
  pointMap <- makeVGAPointMap(
    sfMap,
    gridSize = 0.5,
    fillX = 3.0,
    fillY = 6.0,
    maxVisibility = NA,
```

```
    boundaryGraph = FALSE,
    verbose = FALSE
  )
vgaVisualLocal(pointMap, FALSE)
```

Index