Package 'gridGeometry'

July 22, 2025

Type Package

Title Polygon Geometry in 'grid'

Version 0.4-0

Description Functions for performing polygon geometry with 'grid' grobs. This allows complex shapes to be defined by combining simpler shapes.

URL https://github.com/pmur002/gridgeometry,
https://stattech.wordpress.fos.auckland.ac.nz/2019/03/04/2019-01-a-geometry-engine-interface-for-grid/,
https://stattech.blogs.auckland.ac.nz/2022/06/01/2022-02-constructive-geometry-for-complex-grobs/,
https://stattech.wordpress.fos.auckland.ac.nz/2022/12/14/2022-03-offsetting-lines-and-polygons-in-grid/

Depends R (>= 3.6.0), grid

Imports grDevices, polyclip (>= 1.10-0)

Suggests graphics, lattice

License GPL (>= 2)

NeedsCompilation no

Author Paul Murrell [aut, cre],
Jack Wong [aut]

Maintainer Paul Murrell <paul@stat.auckland.ac.nz>

Repository CRAN

Date/Publication 2024-08-22 10:30:02 UTC

Contents

`grid.minkowski` *Generate Minkowski Sums of Grobs*

Description

Given a polygonal *pattern* and a polygonal *path*, generate the Minkowski Sum by adding the pattern to the path.

Usage

```
minkowskiGrob(A, B,
              grobFn=xyListToPath,
              name=NULL, gp=gpar(), ...)
grid.minkowski(A, B, ...)
```

Arguments

`A`	A grob, gList, or gTree, or a gPath or a character value identifying a grob that has already been drawn. This is known as the *pattern* grob.
`B`	A grob, gList, or gTree, or a gPath or a character value identifying a grob that has already been drawn. This is known as the *path* grob.
`grobFn`	The function that is used to create the final grob result. Predefined options are: `xyListToPath`, `xyListToPolygon`, and `xyListToLine`.
`name`	A name for the resulting grob.
`gp`	Graphical parameter settings for the resulting grob.
`...`	For `minkowskiGrob`, arguments passed on to `polyclip::polyminkowski`.

Details

Both `A` and `B` should not contain self-intersections, though they can be non-convex.

Value

`minkowskiGrob` returns a gTree.

`grid.minkowski` is only used for its side-effect of drawing on the current graphics device.

Author(s)

Jack Wong

See Also

xyListToPath, xyListToPolygon, xyListToLine, polyminkowski

Examples

```
pattern <- circleGrob(x = 0, y = 0, r = .1)
path <- rectGrob(width = 0.5, height = 0.5)
minkowski <- minkowskiGrob(pattern, path)
grid.draw(minkowski)
```

---

grid.polyclip *Perform Geometric Operations on Grobs*

Description

These functions allow two or more grobs to be combined using one of the following operations: intersection, union, minus, and xor.

Usage

```
polyclipGrob(A, B, op="intersection",
             openFn=xyListToLine, closedFn=xyListToPath,
             name=NULL, gp=gpar(),  ...)
grid.polyclip(A, B, ...)
```

Arguments

A	A grob, gList, or gTree, or a gPath or a character value identifying a grob that has already been drawn. This is known as the *subject* grob.
B	A grob, gList, or gTree, or a gPath or a character value identifying a grob that has already been drawn. This is known as the *clip* grob.
op	A character value describing the operation. One of "intersection", "minus", "union", or "xor".
openFn	The function used to create grobs from the open shapes in the result.
closedFn	The function used to create grobs from the closed shapes in the result.
name	A name for the resulting grob.
gp	Graphical parameter settings for the resulting grob.
...	For polyclipGrob, arguments passed on to polyclip. For grid.polyclip, arguments used by methods.

Details

The subject grob is combined with the clip grob using the op operation.

The grobs are converted to coordinates by calling grid::grobCoords and then the operation is performed by calling polyclip.

The result is a new grob. In the case of grid.polyclip, this new grob will be drawn on the current device. In the special case that A is a gPath, by default, the new grob will *replace* the old grob (the original grob identified by A) in the current scene (and the new grob will use the same gp settings as the old grob).

The subject grob can be any combination of open or closed shapes (e.g., a combination of lines and polygons), but the clip grob must only consist of closed shapes.

Value

polyclipGrob returns a gTree with two children, one representing the open shapes within the result and one representing the closed shapes within the result.

grid.polyclip is only used for its side-effect of drawing on the current graphics device.

Author(s)

Paul Murrell

See Also

polyclip

Examples

```
r <- rectGrob(x=1/3, y=1/3, width=.4, height=.4)
c <- circleGrob(x=2/3, y=2/3, r=.2)
grid.draw(r)
grid.draw(c)
grid.polyclip(r, c, gp=gpar(fill="grey"))
```

---

grid.polylineoffset *Perform Line offset region on Grobs*

Description

Given a polygonal lines or open grob, Generate the offset region (guard region, buffer region, morphological dilation) formed by shifting the boundary outwards by a specific distance.

Usage

```
polylineoffsetGrob(A, delta, rule = "winding",
             name=NULL, gp=gpar(),  ...)
grid.polylineoffset(A, delta, ...)
```

Arguments

A	A grob, gList, or gTree, or a gPath or a character value identifying a grob that has already been drawn. This is known as the *subject* grob.
delta	Distance over which the boundary should be shifted.
rule	A character value describing the fill rule to be used. One of "winding", "evenodd"
name	A name for the resulting grob.
gp	Graphical parameter settings for the resulting grob.
...	For polylineoffsetGrob, arguments passed on to polyclip::polylineoffset.

Details

The grobs are converted to coordinates by calling grid::xyListFromGrob.

The result is a new grob. In the case of grid.polylineoffset, this new grob will be drawn on the current device. In the special case that A is a gPath, by default, the new grob will *replace* the old grob (the original grob identified by A) in the current scene (and the new grob will use the same gp settings as the old grob).

The argument jointype determines what happens at the vertices of each line at polyclip::polylineoffset. See code link polylineoffset.

The argument endtype determines what happens at the beginning and end of each line at polyclip::polylineoffset. See code link polylineoffset.

The argument rule is a character value describing the fill rule to be used. One of "winding", "evenodd"

Value

polylineoffsetGrob returns a gTree with offset grob coordinate.

grid.polylineoffset is only used for its side-effect of drawing on the current graphics device.

Author(s)

Jack Wong

See Also

polylineoffset

Examples

```
grobLine = linesGrob(x = c(.4, .8, .8, .2, .6), y = c(.3, .3, .8, .8, .6), name = "Line 1")
grid.polylineoffset(grobLine, delta = unit(0.1, "cm"), jointype="square", endtype = "opensquare")
```

grid.polyoffset *Perform offset region on Grobs*

Description

Given a polygonal region or closed grob, generate the offset region (guard region, buffer region, morphological dilation) formed by shifting the boundary outwards by a specific distance.

Usage

```
polyoffsetGrob(A, delta, reduce = "union", rule = "winding",
             name=NULL, gp=gpar(),  ...)
grid.polyoffset(A, delta, ...)
```

Arguments

A	A grob, gList, or gTree, or a gPath or a character value identifying a grob that has already been drawn. This is known as the *subject* grob.
delta	Distance over which the boundary should be shifted.
rule	A character value describing the fill rule to be used. One of "winding", "evenodd"
reduce	A character value describing the operation to be used if x needs to be collapsed to a single shape. One of "intersection", "flatten", "minus", "union", or "xor".
name	A name for the resulting grob.
gp	Graphical parameter settings for the resulting grob.
...	For polyoffsetGrob, arguments passed on to polyclip::polyoffset.

Details

The grobs are converted to coordinates by calling grid::xyListFromGrob.

The result is a new grob. In the case of grid.polyoffset, this new grob will be drawn on the current device. In the special case that A is a gPath, by default, the new grob will *replace* the old grob (the original grob identified by A) in the current scene (and the new grob will use the same gp settings as the old grob).

Examples of useful arguments that will be passed on to polyclip::polyoffset() are jointype.

The argument jointype determines what happens at the vertices of each line at polyclip::polyoffset. See code link polyoffset.

The argument rule is a character value describing the fill rule to be used. One of "winding", "evenodd"

Value

polyoffsetGrob returns a gTree with offset grob coordinate.

grid.polyoffset is only used for its side-effect of drawing on the current graphics device.

Author(s)

Jack Wong

See Also

polyoffset

Examples

```
grob <- rectGrob(width = 0.5, height = 0.5)
offset <- polyoffsetGrob(grob, 0.2)
grid.draw(offset)
```

---

grid.reduce *Reduce Multiple Shapes to a Single Shape*

Description

This function takes a grob (or gTree) that draws multiple shapes and reduces it to a grob that only draws one shape.

More accurately, the result only draws at most one closed shape and at most one open shape.

Usage

```
reduceGrob(x, op=if (isClosedShape(x)) "union" else "flatten",
           openFn=xyListToLine, closedFn=xyListToPath,
           name=NULL, gp=gpar())
grid.reduce(x, ...)
```

Arguments

x	A grob, gList, or gTree, or a gPath or a character value identifying a grob that has already been drawn.
op	A character value describing the operation. One of "intersection", "minus", "union", "xor", or "flatten".
openFn	The function used to create grobs from the open shapes in the result.
closedFn	The function used to create grobs from the closed shapes in the result.
name	A name for the resulting grob.
gp	Graphical parameter settings for the resulting grob.
...	For reduceGrob, arguments passed on to polyclip. For grid.reduce, arguments used by methods.

Details

Multiple shapes are combined using polyclip() and the specified operator op.

Value

`reduceGrob` returns a gTree with two children, one representing the open shapes within the result and one representing the closed shapes within the result.

`grid.reduce` is only used for its side-effect of drawing on the current graphics device.

Author(s)

Paul Murrell

See Also

`grid.polyclip`

Examples

```
r <- rectGrob(x=1/3, y=1/3, width=.4, height=.4)
c <- circleGrob(x=2/3, y=2/3, r=.3)
grid.reduce(grobTree(r, c), gp=gpar(fill="grey"))
```

---

grid.trim *Subset a Line*

Description

These functions generate one or more lines by subsetting a line.

Usage

```
trimGrob(x, from, to, rep=FALSE, name=NULL, gp=gpar(), ...)
grid.trim(x, ...)
```

Arguments

`x`	A grob, or a gPath (or a character value) identifying a grob that has already been drawn.
`from`	A numeric vector or a unit object describing the start point of each subset.
`to`	A numeric vector or a unit object describing the end point of each subset.
`rep`	A logical value indicating whether the `from` and `to` values should be repeated to consume the entire line.
`name`	A name for the resulting grob.
`gp`	Graphical parameter settings for the resulting grob.
`...`	For `trimGrob`, arguments passed on to `trim`. For `grid.trim`, arguments used by methods.

Details

Both `from` and `to` should be either numeric values between 0 and 1, expressing a proportion of the line length, or unit objects. In the latter case, the unit is converted to a proportion of the line length, with `"npc"` units treated as proportions of the line length.

Both `from` and `to` can be vectors (and they are recycled) so that multiple subsets can be obtained in a single call.

Both `from` and `to` can be negative, in which case the value is treated as distance backwards along the line from its end.

The result is a new grob. In the case of `grid.trim`, this new grob will be drawn on the current device. In the special case that `x` is a gPath, by default, the new grob will *replace* the old grob (the original grob identified by `x`) in the current scene (and the new grob will use the same `gp` settings as the old grob).

Value

`trimGrob` returns a polyline grob.

`grid.trim` is only used for its side-effect of drawing on the current graphics device.

Author(s)

Paul Murrell

See Also

`trim`

Examples

```
g <- xsplineGrob(c(.2, .5, .8), c(.2, .8, .2))
grid.draw(g)
grid.trim(g, from=.1, to=.2, gp=gpar(lwd=5))
grid.trim(g, from=.1, to=.2, rep=TRUE, gp=gpar(lwd=3))
```

---

polyclip *Perform Geometric Operations on Coordinates*

Description

This function combines two sets of coordinates using one of the following operations: intersection, union, minus, and xor.

Usage

```
polyclip(A, B, ...)
## S3 method for class 'grob'
polyclip(A, B, op="intersection", closed=isClosedShape(A),
         reduceA = if (closed) "union" else "flatten",
         reduceB = "union",
         fillA = NULL, fillB = NULL,
         ...)
## S3 method for class 'gList'
polyclip(A, B, op="intersection", closed=isClosedShape(A),
         reduceA = if (closed) "union" else "flatten",
         reduceB = "union",
         fillA = NULL, fillB = NULL,
         ...)
## S3 method for class 'gPath'
polyclip(A, B, op="intersection", closed,
                        strict=FALSE, grep=FALSE, global=FALSE,
         reduceA = if (closed) "union" else "flatten",
         reduceB = "union",
         fillA = NULL, fillB = NULL,
         ...)
## S3 method for class 'character'
polyclip(A, B, op="intersection", closed,
                             strict=FALSE, grep=FALSE, global=FALSE,
         reduceA = if (closed) "union" else "flatten",
         reduceB = "union",
         fillA = NULL, fillB = NULL,
         ...)
```

Arguments

A	A set of coordinates describing the *subject* shape. Or a grob, gList, or a gPath (or a character value) identifying a grob that has already been drawn from which coordinates are generated.
B	A set of coordinates describing the *clip* shape. Or a grob, gList, or a gPath (or a character value) identifying a grob that has already been drawn from which coordinates are generated.
op	A character value describing the operation. One of "intersection", "minus", "union", or "xor".
closed	A logical value indicating whether the A coordinates describe a closed shape or an open shape.
reduceA, reduceB	A character value describing the operation to be used if either A or B need to be reduced to a single set of coordinates. One of "intersection", "minus", "union", or "xor", in which case polyclip is used to reduce multiple shapes, or "flatten", in which case coordinates for all shapes are returned.

`fillA`, `fillB` A character value describing the fill rule. Possible values are `"winding"` or `"evenodd"` or `NULL`. In the latter case (the default), the fill rule will be taken from the coordinates if the coordinates have a `"rule"` attribute. Otherwise, the default is `"winding"`.

`strict`, `grep`, `global` Arguments controlling the interpretation of the gPath (passed to `grid.get`).

`...` Arguments used by methods.

Details

The subject coordinates are combined with the clip coordinates using the `op` operation.

Value

The result is a new set of coordinates.

Author(s)

Paul Murrell

See Also

`polyclip`

Examples

```
r <- rectGrob(x=1/3, y=1/3, width=.4, height=.4)
c <- circleGrob(x=2/3, y=2/3, r=.2)
polyclip(r, c)
```

---

polylineoffset *Perform Line offset region on Grobs*

Description

Given a list of polygonal lines or open grob object, generate the offset region (guard region, buffer region, morphological dilation) formed by shifting the boundary outwards by a specific distance.

Usage

```
polylineoffset(A, delta, ...)
## S3 method for class 'grob'
polylineoffset(A, delta,
         ...)
## S3 method for class 'gList'
polylineoffset(A, delta,
         ...)
## S3 method for class 'gPath'
```

```
polylineoffset(A, delta,
                          ..., strict=FALSE, grep=FALSE, global=FALSE)
## S3 method for class 'character'
polylineoffset(A, delta,
                             ..., strict=FALSE, grep=FALSE, global=FALSE)
## S3 method for class 'list'
polylineoffset(A, delta,
                             ...)
```

Arguments

A	A set of coordinates describing the *subject* shape. Or a grob, gList, or a gPath (or a character value) identifying a grob that has already been drawn from which coordinates are generated.
delta	Distance over which the boundary should be shifted.
strict, grep, global	Arguments controlling the interpretation of the gPath (passed to grid.get).
...	For polylineoffsetGrob, arguments passed on to polyclip::polylineoffset.

Details

Calculate the offset region by using the subject coordinates shift by a delta distance.

The argument jointype determines what happens at the vertices of each line.

- jointype = "round": a circular arc is generated.
- jointype = "square": circular arc is replaced by a single straight line.
- jointype = "miter": circular arc is omitted entirely and replaced by a single straight line.

The argument endtype determines what happens at the beginning and end of each line.

- endtype = "closedpolygon": ends are joined together (using the jointype value) and the path filled as a polygon.
- endtype = "closedline": nds are joined together (using the jointype value) and the path filled as a polyline.
- endtype = "openbutt": ends are squared off abruptly.
- endtype = "opensquare": ends are squared off at distance delta.
- endtype = "openround": ends are replaced by a semicircular arc.

Value

The result is a new set of coordinates for the outline of the offset region.

Author(s)

Jack Wong

See Also

polylineoffset

Examples

```
grobLine = linesGrob(x = c(.4, .8, .8, .2, .6), y = c(.3, .3, .8, .8, .6), name = "Line 1")
offset <- polylineoffset(grobLine, delta = unit(0.1, "cm"),
                         jointype="square", endtype = "opensquare")
```

---

polyminkowski *Generate Minkowski Sums on Coordinates*

Description

This function generates the Minkowski Sum of two sets of coordinates.

Usage

```
polyminkowski(A, B, ...)
## S3 method for class 'grob'
polyminkowski(A, B, closed=isClosedShape(B),
         reduceA = "union",
         reduceB = "union",
         ...)
## S3 method for class 'gList'
polyminkowski(A, B, closed=isClosedShape(B),
         reduceA = "union",
         reduceB = "union",
         ...)
## S3 method for class 'gPath'
polyminkowski(A, B, closed,
                         strict=FALSE, grep=FALSE, global=FALSE,
         reduceA = "union",
         reduceB = "union",
         ...)
## S3 method for class 'character'
polyminkowski(A, B, closed,
                              strict=FALSE, grep=FALSE, global=FALSE,
         reduceA = "union",
         reduceB = "union",
         ...)
```

Arguments

A	A set of coordinates describing a *pattern* shape. Or a grob, gList, or a gPath (or a character value) identifying a grob that has already been drawn from which coordinates are generated.

B	A set of coordinates describing a *path* shape. Or a grob, gList, or a gPath (or a character value) identifying a grob that has already been drawn from which coordinates are generated.
closed	A logical value indicating whether the B coordinates describe a closed shape or an open shape.
reduceA, reduceB	A character value describing the operation to be used if either A or B need to be reduced to a single set of coordinates. One of "intersection", "minus", "union", or "xor", in which case polyminkowski is used to reduce multiple shapes, or "flatten", in which case coordinates for all shapes are returned.
strict, grep, global	Arguments controlling the interpretation of the gPath (passed to grid.get).
...	Arguments used by methods.

Details

The shape described by the pattern coordinates is added to the shape described by the path coordinates.

Value

The result is a new set of coordinates.

Author(s)

Paul Murrell

See Also

grid.minkowski

Examples

```
c <- circleGrob(x=0, y=0, r=.1)
r <- rectGrob(width=.5, height=.5)
polyminkowski(c, r)
```

---

polyoffset *Perform offset region on Grobs*

Description

Given a polygonal region or closed grob, generate the offset region (guard region, buffer region, morphological dilation) formed by shifting the boundary outwards by a specific distance.

Usage

```
polyoffset(A, delta, reduce = "union", ...)
## S3 method for class 'grob'
polyoffset(A, delta, reduce = "union",
         ...)
## S3 method for class 'gList'
polyoffset(A, delta, reduce = "union",
         ...)
## S3 method for class 'gPath'
polyoffset(A, delta, reduce = "union",
                         ..., strict=FALSE, grep=FALSE, global=FALSE)
## S3 method for class 'character'
polyoffset(A, delta, reduce = "union",
                            ..., strict=FALSE, grep=FALSE, global=FALSE)
## S3 method for class 'list'
polyoffset(A, delta, reduce = "union",
                            ...)
```

Arguments

A	A set of coordinates describing the *subject* shape. Or a grob, gList, or a gPath (or a character value) identifying a grob that has already been drawn from which coordinates are generated.
delta	Distance over which the boundary should be shifted.
reduce	A character value describing the operation to be used if A needs to be collapsed to a single shape. One of "intersection", "flatten", "minus", "union", or "xor".
strict, grep, global	Arguments controlling the interpretation of the gPath (passed to grid.get).
...	For polyoffsetGrob, arguments passed on to polyclip::polyoffset.

Details

Calculate the offset region by using the subject coordinates shift by a delta distance.

Examples of useful arguments that will be passed on to polyclip::polyoffset() are jointype.

The argument jointype determines what happens at the vertices of each line.

- jointype = "round": a circular arc is generated.
- jointype = "square": circular arc is replaced by a single straight line.
- jointype = "miter": circular arc is omitted entirely and replaced by a single straight line.

Value

The result is a new set of coordinates for the outline of the offset region.

Author(s)

Jack Wong

See Also

polyoffset

Examples

```
grob <- rectGrob(width = 0.5, height = 0.5)
offset <- polyoffset(grob, 0.2)
```

---

trim *Generate Subsets of Coordinates*

Description

This functions generates a new set of coordinates by subsetting a set of coordinates.

Usage

```
trim(x, from, to, ...)
## S3 method for class 'grob'
trim(x, from, to, rep=FALSE, ...)
## S3 method for class 'gPath'
trim(x, from, to, rep=FALSE,
                    strict=FALSE, grep=FALSE, global=FALSE, ...)
## S3 method for class 'character'
trim(x, from, to, rep=FALSE,
                        strict=FALSE, grep=FALSE, global=FALSE, ...)
```

Arguments

x	A set of coordinates. Or a grob, or a gPath (or a character value) identifying a grob that has already been drawn from which coordinates are generated.
from	A numeric vector or a unit object describing the start point of each subset.
to	A numeric vector or a unit object describing the end point of each subset.
rep	A logical value indicating whether the from and to values should be repeated to consume the entire line.
strict, grep, global	Arguments controlling the interpretation of the gPath (passed to grid.get).
...	Arguments used by methods.

Value

A new set of coordinates.

Author(s)

Paul Murrell

See Also

trim

Examples

```
g <- segmentsGrob(0, .5, 1, .5)
trim(g, from=.1, to=.2)
trim(g, from=.1, to=.2, rep=TRUE)
```

---

xyListFromGrob *Generate Coordinates from Grobs*

Description

This function generates a set of coordinates (a list of (x,y) lists) from a grob.

Usage

```
xyListFromGrob(x, op = if (closed) "union" else "flatten",
               closed = isClosedShape(x), ...)
```

Arguments

x	A grob.
op	A character value describing the operation to be used if x needs to be collapsed to a single shape. One of "intersection", "minus", "union", or "xor". Can also be "flatten", which just returns an (x, y) list for each shape the grob draws. This is usually the best choice when closed = FALSE.
closed	A logical value indicating whether we coordinates describing a closed shape or an open shape from.
...	Arguments passed on to polyclip.

Value

The result is a list of lists, each with components x and y.

Author(s)

Paul Murrell

See Also

polyclip

Examples

```
r <- rectGrob(x=1/3, y=1/3, width=.4, height=.4)
xyListFromGrob(r)
```

xyListPath *Generate Grobs from Coordinates*

Description

This function generates a grob from a set of coordinates (a list of (x,y) lists).

Usage

```
xyListToPath(x, rule=, name=NULL, gp=gpar())
xyListToPolygon(x, name=NULL, gp=gpar())
xyListToLine(x, name=NULL, gp=gpar())

xyListPath(x, rule, name=NULL, gp=gpar())
xyListPolygon(x, name=NULL, gp=gpar())
xyListLine(x, name=NULL, gp=gpar())
```

Arguments

x	A set of coordinates (e.g., from `polyclip` or `trim`).
rule	A fill rule: `"winding"` or `"evenodd"`.
name	A name for the resulting grob.
gp	Graphical parameter settings for the resulting grob.

Details

Following the addition of the `xyListFromGrob` function, the `*To*()` forms are preferred.

Value

The result is a grob.

Author(s)

Paul Murrell

See Also

`polyclip`

Examples

```
r <- rectGrob(x=1/3, y=1/3, width=.4, height=.4)
c <- circleGrob(x=2/3, y=2/3, r=.2)
coords <- polyclip(r, c)
xyListPath(coords)
xyListLine(coords)
```

Index