line-intersect-2d

Release 1.2.1a1

Piotr Maślanka

Dec 16, 2020

CONTENTS:

CHAPTER

ONE

BASICS

The basic classes are as follows:

class line_intersect_2d.basics.**Point**

A single point.

This is immutable, hashable and __eq__able. Take care when comparing floats.

This overloads +, -, * and /

Parameters

- **x** (*float*) – x coordinate
- **y** (*float*) – y coordinate

Variables

- **x** – x coordinate (float)
- **y** – y coordinate (float)

add()

Returns result of adding this point to another point

Parameters **p** (Point) – point p

Returns new Point

Return type *Point*

div()

Returns result of dividing this point by a factor

Parameters **p** (*float*) – point p

Returns new Point

Return type *Point*

mul()

Returns result of multiplying this point by a factor

Parameters **p** (*float*) – point p

Returns new Point

Return type *Point*

sub()

Returns result of the difference between this point and p

Parameters **p** (Point) – point p

Returns new Point

Return type *Point*

class line_intersect_2d.basics.**Segment**

A segment.

This is immutable (save for tag), __eq__able and hashable.

Parameters

- **start** (*Vector*) – start point
- **stop** (*Vector*) – stop point

Variables

- **start** – start point (Point)
- **stop** – stop point (Point)
- **tag** – tag (int), writable
- **q_nodes** – numbers of q-nodes that this segment belongs to (tp.List[int])

intersection_point()

Get the point of intersection between this segment and s

Parameters **sa** (Segment) – segment s

Returns point of intersection

Return type *Point*

Raises **ValueError** – there is no intesection

class line_intersect_2d.quadtrees.**Path**

A path made from connected segments.

This is immutable.

Constructor works as:

```
>>> p = Path([Segment(...), Segment(...)])
```

or

```
>>> p = Path((x1, y1), (x2, y2), ...)
```

Variables **segments** – list of segments (tp.List[Segment])

USAGE

First you need to create your `Path` objects. Assume that paths you pass are numbered from 0 to n.

After you make them, you just pass them to

`line_intersect_2d.quadtrees.`**`check_intersection`**`()`

Check whether any number of paths intersect.

Parameters

- **paths** (*tp.List[Path]*) – paths to check
- **split_factor** (*float*) – Factor that the tree should be constructed. Eg. for the default value of 0.1 the grid will be divided into 10 rows and 10 columns. Default is 0.1

Returns a tuple of two segments from different paths that intersect, or None if no intersection

Return type tp.Optional[tp.Tuple[*Segment*, *Segment*]]

Note that a *split_factor* will divide the grid into *(1/split_factor)**2*, so in case of the default *split_factor* of 0.1 100 subrectangles will be made.

Which will return either a tuple of (`Segment`, `Segment`) two segments from different paths (which paths it will be stored in their `tag` attribute, the number that was aforementioned) or *None* will be returned, if they don't collide

You can use later `line_intersect_2d.basics.Segment.intersection_point()` to calculate the intersection point.

Installation:

Just do

```
pip install snakehouse satella
pip install line-intersect-2d
```

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

INDEX