

## CRoPS (Combined Roadmap and Potentials for Swarms)

Generated by Doxygen 1.8.4

Sun Dec 8 2013 00:48:05



# Contents

<b>1</b>	<b>Namespace Index</b>	<b>1</b>
1.1	Namespace List . . . . .	1
<b>2</b>	<b>Hierarchical Index</b>	<b>3</b>
2.1	Class Hierarchy . . . . .	3
<b>3</b>	<b>Class Index</b>	<b>5</b>
3.1	Class List . . . . .	5
<b>4</b>	<b>File Index</b>	<b>7</b>
4.1	File List . . . . .	7
<b>5</b>	<b>Namespace Documentation</b>	<b>9</b>
5.1	boid Namespace Reference . . . . .	9
5.1.1	Function Documentation . . . . .	9
5.1.1.1	guassianFunc . . . . .	9
5.1.2	Variable Documentation . . . . .	10
5.1.2.1	__author__ . . . . .	10
5.2	boidsimulation Namespace Reference . . . . .	10
5.2.1	Variable Documentation . . . . .	10
5.2.1.1	__author__ . . . . .	10
5.3	configuration Namespace Reference . . . . .	10
5.4	dijkstra Namespace Reference . . . . .	10
5.4.1	Function Documentation . . . . .	11
5.4.1.1	Dijkstra . . . . .	11
5.4.1.2	shortestPath . . . . .	12
5.5	gatherstats Namespace Reference . . . . .	13
5.5.1	Function Documentation . . . . .	13
5.5.1.1	generateStats . . . . .	13
5.5.2	Variable Documentation . . . . .	13
5.5.2.1	mapList . . . . .	13
5.5.2.2	testList . . . . .	13
5.6	goal Namespace Reference . . . . .	13

5.6.1	Variable Documentation	14
5.6.1.1	__author__	14
5.7	mapparser Namespace Reference	14
5.7.1	Function Documentation	14
5.7.1.1	mapVal	14
5.7.1.2	mparse	14
5.8	obstacle Namespace Reference	15
5.8.1	Variable Documentation	15
5.8.1.1	__author__	15
5.9	pridict Namespace Reference	15
5.10	prm Namespace Reference	15
5.11	test_sim Namespace Reference	16
5.11.1	Variable Documentation	16
5.11.1.1	arg	16
5.11.1.2	dynamicObstacleAutoGenerate	16
5.11.1.3	endPoint	16
5.11.1.4	flockSize	16
5.11.1.5	fs	16
5.11.1.6	generateTarget	16
5.11.1.7	mapDict	17
5.11.1.8	mapFilePath	17
5.11.1.9	obstacleFilePath	17
5.11.1.10	startPoint	17
<b>6</b>	<b>Class Documentation</b>	<b>19</b>
6.1	boid.Boid Class Reference	19
6.1.1	Detailed Description	22
6.1.2	Constructor & Destructor Documentation	22
6.1.2.1	__init__	22
6.1.3	Member Function Documentation	22
6.1.3.1	determineNewPath	22
6.1.3.2	determineRandomWalk	22
6.1.3.3	draw	23
6.1.3.4	findMax	23
6.1.3.5	getBoidVectorList	23
6.1.3.6	getDirectionVector	24
6.1.3.7	getGoalVector	24
6.1.3.8	getNeighborVectorList	25
6.1.3.9	getObstacleVectorList	26
6.1.3.10	getVar	27

6.1.3.11	<a href="#">inGoal</a>	27
6.1.3.12	<a href="#">initFunctionParameters</a>	28
6.1.3.13	<a href="#">inWorld</a>	28
6.1.3.14	<a href="#">mag</a>	28
6.1.3.15	<a href="#">norm</a>	29
6.1.3.16	<a href="#">obstacleFunc</a>	29
6.1.3.17	<a href="#">pointAllowed</a>	30
6.1.3.18	<a href="#">reduceWeightValues</a>	30
6.1.3.19	<a href="#">setBoidList</a>	31
6.1.3.20	<a href="#">setNewGoal</a>	31
6.1.3.21	<a href="#">sigmoidFunc</a>	31
6.1.3.22	<a href="#">sumDivide</a>	32
6.1.3.23	<a href="#">update</a>	33
6.1.3.24	<a href="#">updatePositionBuffer</a>	33
6.1.4	<a href="#">Member Data Documentation</a>	33
6.1.4.1	<a href="#">bAlpha</a>	33
6.1.4.2	<a href="#">bBeta</a>	33
6.1.4.3	<a href="#">bConst</a>	33
6.1.4.4	<a href="#">bDelta</a>	34
6.1.4.5	<a href="#">bInfluenceR</a>	34
6.1.4.6	<a href="#">boidList</a>	34
6.1.4.7	<a href="#">color</a>	34
6.1.4.8	<a href="#">compWeightList</a>	34
6.1.4.9	<a href="#">dim</a>	34
6.1.4.10	<a href="#">endIndex</a>	34
6.1.4.11	<a href="#">ePos</a>	34
6.1.4.12	<a href="#">gAlpha</a>	34
6.1.4.13	<a href="#">gammaFunc</a>	34
6.1.4.14	<a href="#">gBeta</a>	35
6.1.4.15	<a href="#">gConst</a>	35
6.1.4.16	<a href="#">gDelta</a>	35
6.1.4.17	<a href="#">goal</a>	35
6.1.4.18	<a href="#">goalCounter</a>	35
6.1.4.19	<a href="#">goalList</a>	35
6.1.4.20	<a href="#">goalNodes</a>	35
6.1.4.21	<a href="#">heading</a>	35
6.1.4.22	<a href="#">headWeightList</a>	35
6.1.4.23	<a href="#">neighborSize</a>	36
6.1.4.24	<a href="#">nStuckDAvg</a>	36
6.1.4.25	<a href="#">nStuckDSigma</a>	36

6.1.4.26	obBeta	36
6.1.4.27	obInfluenceR	36
6.1.4.28	obstacleList	36
6.1.4.29	position	36
6.1.4.30	positionBuffer	36
6.1.4.31	prmGen	36
6.1.4.32	radius	37
6.1.4.33	randomWalkX	37
6.1.4.34	randomWalkY	37
6.1.4.35	randWalkCount	37
6.1.4.36	roadmap	37
6.1.4.37	screen	37
6.1.4.38	speed	37
6.1.4.39	sPos	37
6.1.4.40	stuck	37
6.1.4.41	stuckConst	38
6.1.4.42	stuckDAvg	38
6.1.4.43	stuckDSigma	38
6.1.4.44	ySize	38
6.2	goal.CircleGoal Class Reference	38
6.2.1	Detailed Description	39
6.2.2	Constructor & Destructor Documentation	39
6.2.2.1	__init__	39
6.2.3	Member Function Documentation	39
6.2.3.1	draw	39
6.2.4	Member Data Documentation	39
6.2.4.1	colors	39
6.2.4.2	position	39
6.2.4.3	radius	39
6.2.4.4	screen	39
6.3	configuration.Configuration Class Reference	40
6.3.1	Detailed Description	41
6.3.2	Member Data Documentation	41
6.3.2.1	boidSpeed	41
6.3.2.2	colorList	41
6.3.2.3	dim	41
6.3.2.4	goalRadius	41
6.3.2.5	numNeighbours	41
6.3.2.6	numSamplePoints	41
6.3.2.7	screen	42

6.4	dict Class Reference	42
6.5	boidsimulation.FlockSim Class Reference	42
6.5.1	Detailed Description	43
6.5.2	Constructor & Destructor Documentation	44
6.5.2.1	__init__	44
6.5.3	Member Function Documentation	44
6.5.3.1	animate	44
6.5.3.2	avg	44
6.5.3.3	getBoidData	45
6.5.3.4	getStats	45
6.5.3.5	init_prm	45
6.5.3.6	play	46
6.5.3.7	render	46
6.5.4	Member Data Documentation	47
6.5.4.1	auto_gen_number	47
6.5.4.2	auto_gen_obst	47
6.5.4.3	BLACK	47
6.5.4.4	config	47
6.5.4.5	counter	47
6.5.4.6	dataFile	47
6.5.4.7	dim	47
6.5.4.8	done	48
6.5.4.9	ePos	48
6.5.4.10	flockSize	48
6.5.4.11	font	48
6.5.4.12	frameCounter	48
6.5.4.13	iterations	48
6.5.4.14	mapFile	48
6.5.4.15	numInGoal	48
6.5.4.16	obstacleFile	48
6.5.4.17	sPos	49
6.5.4.18	startTime	49
6.5.4.19	surfaceList	49
6.5.4.20	WHITE	49
6.6	configuration.PolyFileConfiguration Class Reference	49
6.6.1	Detailed Description	51
6.6.2	Member Function Documentation	51
6.6.2.1	autoGenerateDynamicObstacles	51
6.6.2.2	initVars	51
6.6.2.3	parseDynamicObstacles	51

6.6.3	Member Data Documentation . . . . .	51
6.6.3.1	auto_gen_number . . . . .	51
6.6.3.2	auto_gen_obst . . . . .	51
6.6.3.3	boidList . . . . .	51
6.6.3.4	endPoint . . . . .	52
6.6.3.5	goalList . . . . .	52
6.6.3.6	nodes . . . . .	52
6.6.3.7	obstacleList . . . . .	52
6.6.3.8	prmGen . . . . .	52
6.6.3.9	startPoint . . . . .	52
6.7	obstacle.PolyObstacle Class Reference . . . . .	52
6.7.1	Detailed Description . . . . .	54
6.7.2	Constructor & Destructor Documentation . . . . .	54
6.7.2.1	__init__ . . . . .	54
6.7.3	Member Function Documentation . . . . .	54
6.7.3.1	change_direction . . . . .	54
6.7.3.2	checkCollisionWithOtherObstacles . . . . .	54
6.7.3.3	detectCollision . . . . .	54
6.7.3.4	determine_last_direction . . . . .	55
6.7.3.5	draw . . . . .	55
6.7.3.6	estimatePoly . . . . .	55
6.7.3.7	getClosestPoint . . . . .	55
6.7.3.8	getPoint . . . . .	56
6.7.3.9	getRadius . . . . .	56
6.7.3.10	norm . . . . .	57
6.7.3.11	pointAllowed . . . . .	57
6.7.3.12	pointInPoly . . . . .	58
6.7.3.13	rayintersectseg . . . . .	59
6.7.3.14	removeSelfFromObstacleList . . . . .	59
6.7.3.15	translate . . . . .	59
6.7.4	Member Data Documentation . . . . .	60
6.7.4.1	avgPoint . . . . .	60
6.7.4.2	boundary . . . . .	60
6.7.4.3	colors . . . . .	60
6.7.4.4	displacement . . . . .	60
6.7.4.5	dynamic . . . . .	60
6.7.4.6	max_displacement . . . . .	60
6.7.4.7	maxDist . . . . .	60
6.7.4.8	nodes . . . . .	60
6.7.4.9	obstacles . . . . .	60



6.7.4.10	screen	61
6.7.4.11	velocity	61
6.8	pridict.priorityDictionary Class Reference	61
6.8.1	Detailed Description	62
6.8.2	Constructor & Destructor Documentation	62
6.8.2.1	__init__	62
6.8.3	Member Function Documentation	62
6.8.3.1	__iter__	62
6.8.3.2	__setitem__	62
6.8.3.3	setdefault	63
6.8.3.4	smallest	63
6.9	prm.PRMGenerator Class Reference	63
6.9.1	Detailed Description	64
6.9.2	Constructor & Destructor Documentation	64
6.9.2.1	__init__	64
6.9.3	Member Function Documentation	65
6.9.3.1	draw	65
6.9.3.2	drawPath	65
6.9.3.3	filterSubGoal	65
6.9.3.4	findNeighbors	65
6.9.3.5	generate	66
6.9.3.6	generatePositionList	66
6.9.3.7	getRandom	67
6.9.3.8	getShortestPath	67
6.9.3.9	initOmega	67
6.9.3.10	norm	68
6.9.4	Member Data Documentation	68
6.9.4.1	adjacentThresh	68
6.9.4.2	dontDraw	69
6.9.4.3	endPos	69
6.9.4.4	goalNodes	69
6.9.4.5	gPosList	69
6.9.4.6	numNext	69
6.9.4.7	obstacleList	69
6.9.4.8	omegaDict	69
6.9.4.9	roadmap	69
6.9.4.10	screen	69
6.9.4.11	startPos	70
6.9.4.12	subGoalNumber	70
6.9.4.13	subGoalPositionList	70

6.9.4.14	xSize	70
6.9.4.15	ySize	70
<b>7</b>	<b>File Documentation</b>	<b>71</b>
7.1	code/boid.py File Reference	71
7.2	code/boidsimulation.py File Reference	71
7.3	code/configuration.py File Reference	72
7.4	code/dijkstra.py File Reference	72
7.5	code/gatherstats.py File Reference	72
7.6	code/goal.py File Reference	73
7.7	code/mapparser.py File Reference	73
7.8	code/obstacle.py File Reference	74
7.9	code/priodict.py File Reference	74
7.10	code/prm.py File Reference	74
7.11	code/test_sim.py File Reference	75
<b>Index</b>		<b>76</b>

# Chapter 1

## Namespace Index

### 1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">boid</a>	9
<a href="#">boidsimulation</a>	10
<a href="#">configuration</a>	10
<a href="#">dijkstra</a>	10
<a href="#">gatherstats</a>	13
<a href="#">goal</a>	13
<a href="#">mapparser</a>	14
<a href="#">obstacle</a>	15
<a href="#">pridict</a>	15
<a href="#">prm</a>	15
<a href="#">test_sim</a>	16



## Chapter 2

# Hierarchical Index

### 2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

boid.Boid . . . . .	19
goal.CircleGoal . . . . .	38
configuration.Configuration . . . . .	40
configuration.PolyFileConfiguration . . . . .	49
dict . . . . .	42
pridict.priorityDictionary . . . . .	61
boidsimulation.FlockSim . . . . .	42
obstacle.PolyObstacle . . . . .	52
prm.PRMGenerator . . . . .	63



## Chapter 3

# Class Index

### 3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">boid.Boid</a>	Class which represents one boid . . . . .	19
<a href="#">goal.CircleGoal</a>	Object that holds data about a goal modelled as circular configuration space . . . . .	38
<a href="#">configuration.Configuration</a>	Static class that holds important global variables . . . . .	40
<a href="#">dict</a>	. . . . .	42
<a href="#">boidsimulation.FlockSim</a>	Main class for that is used for the simulation and display of the flock . . . . .	42
<a href="#">configuration.PolyFileConfiguration</a>	Extends the <a href="#">Configuration</a> class . . . . .	49
<a href="#">obstacle.PolyObstacle</a>	Object that represents the an obstacle represented by a series of points (in the node list) which make up a set of lines . . . . .	52
<a href="#">priodict.priorityDictionary</a>	. . . . .	61
<a href="#">prm.PRMGenerator</a>	Class used to hold methods and variables that are important for the global path planning problem	63





## Chapter 4

# File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

<a href="#">code/boid.py</a>	71
<a href="#">code/boidsimulation.py</a>	71
<a href="#">code/configuration.py</a>	72
<a href="#">code/dijkstra.py</a>	72
<a href="#">code/gatherstats.py</a>	72
<a href="#">code/goal.py</a>	73
<a href="#">code/mapparser.py</a>	73
<a href="#">code/obstacle.py</a>	74
<a href="#">code/priodict.py</a>	74
<a href="#">code/prm.py</a>	74
<a href="#">code/test_sim.py</a>	75



## Chapter 5

# Namespace Documentation

### 5.1 boid Namespace Reference

#### Classes

- class [Boid](#)

*Class which represents one boid.*

#### Functions

- def [guassianFunc](#)

*Gamma function used to give a probability distribution of the flock in order to choose an appropriate neighbour.*

#### Variables

- string [\\_\\_author\\_\\_](#) = "Alex Wallar <aw204@st-andrews.ac.uk>"

#### 5.1.1 Function Documentation

5.1.1.1 `def boid.guassianFunc ( dX, dAvg = 10, dSigma = 1 )`

Gamma function used to give a probability distribution of the flock in order to choose an appropriate neighbour.

##### Parameters

<i>dX</i>	Distance from the boid to the prospective neighbour
<i>dAvg</i>	Dynamic variable used to define the average prospective Distance
<i>dSigma</i>	Standard deviation of the average distances

Definition at line 21 of file boid.py.

Here is the caller graph for this function:



## 5.1.2 Variable Documentation

5.1.2.1 string `boid.__author__` = "Alex Wallar <aw204@st-andrews.ac.uk>"

Definition at line 3 of file `boid.py`.

## 5.2 boidsimulation Namespace Reference

### Classes

- class [FlockSim](#)  
*Main class for that is used for the simulation and display of the flock.*

### Variables

- string `__author__` = "Alex Wallar <aw204@st-andrews.ac.uk>"

## 5.2.1 Variable Documentation

5.2.1.1 string `boidsimulation.__author__` = "Alex Wallar <aw204@st-andrews.ac.uk>"

Definition at line 4 of file `boidsimulation.py`.

## 5.3 configuration Namespace Reference

### Classes

- class [Configuration](#)  
*Static class that holds important global variables.*
- class [PolyFileConfiguration](#)  
*Extends the [Configuration](#) class.*

## 5.4 dijkstra Namespace Reference

### Functions

- def [Dijkstra](#)

*Find shortest paths from the start vertex to all vertices nearer than or equal to the end.*

- def [shortestPath](#)

*Find a single shortest path from the given start vertex to the given end vertex.*

### 5.4.1 Function Documentation

#### 5.4.1.1 `def dijkstra.Dijkstra ( G, start, end=None )`

Find shortest paths from the start vertex to all vertices nearer than or equal to the end.

The input graph `G` is assumed to have the following representation: A vertex can be any object that can be used as an index into a dictionary. `G` is a dictionary, indexed by vertices. For any vertex `v`, `G[v]` is itself a dictionary, indexed by the neighbors of `v`. For any edge `v->w`, `G[v][w]` is the length of the edge. This is related to the representation in <http://www.python.org/doc/essays/graphs.html> where Guido van Rossum suggests representing graphs as dictionaries mapping vertices to lists of neighbors, however dictionaries of edges have many advantages over lists: they can store extra information (here, the lengths), they support fast existence tests, and they allow easy modification of the graph by edge insertion and removal. Such modifications are not needed here but are important in other graph algorithms. Since dictionaries obey iterator protocol, a graph represented as described here could be handed without modification to an algorithm using Guido's representation.

Of course, `G` and `G[v]` need not be Python dict objects; they can be any other object that obeys dict protocol, for instance a wrapper in which vertices are URLs and a call to `G[v]` loads the web page and finds its links.

The output is a pair `(D,P)` where `D[v]` is the distance from start to `v` and `P[v]` is the predecessor of `v` along the shortest path from `s` to `v`.

Dijkstra's algorithm is only guaranteed to work correctly when all edge lengths are positive. This code does not verify this property for all edges (only the edges seen before the end vertex is reached), but will correctly compute shortest paths even for some graphs with negative edges, and will raise an exception if it discovers that a negative edge has caused it to make a mistake.

#### Parameters

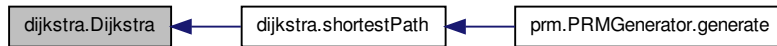
<code>G</code>	The graph dictionary to be searched
<code>start</code>	Starting node
<code>end</code>	End node

**Returns**

Something important

Definition at line 53 of file dijkstra.py.

Here is the caller graph for this function:

**5.4.1.2 def dijkstra.shortestPath ( *G*, *start*, *end* )**

Find a single shortest path from the given start vertex to the given end vertex.

The input has the same conventions as `Dijkstra()`.  
 The output is a list of the vertices in order along  
 the shortest path.

**Parameters**

<i>G</i>	The graph dictionary to be searched
<i>start</i>	The starting node
<i>end</i>	The ending node

**Returns**

A list of the nodes that lie on the shortest path from start to end in G

Definition at line 89 of file dijkstra.py.

Here is the call graph for this function:



Here is the caller graph for this function:



## 5.5 gatherstats Namespace Reference

### Functions

- def [generateStats](#)  
*Generates the statistics and saves them in a well known location.*

### Variables

- list [mapList](#)  
*A list of dictionaries used to store the map files, starting and ending points of the boids.*
- [testList](#) = [mapList](#)

#### 5.5.1 Function Documentation

##### 5.5.1.1 def gatherstats.generateStats ( *mapFile*, *iterations*, *startPoint*, *endPoint* )

Generates the statistics and saves them in a well known location.

##### Parameters

<i>mapFile</i>	The file that stores information about the environment
<i>iterations</i>	the number of experiments to run per number of bots
<i>startPoint,end-Point</i>	Defines the starting and ending points of the flock

Definition at line 14 of file gatherstats.py.

#### 5.5.2 Variable Documentation

##### 5.5.2.1 list gatherstats.mapList

A list of dictionaries used to store the map files, starting and ending points of the boids.

Definition at line 47 of file gatherstats.py.

##### 5.5.2.2 gatherstats.testList = mapList

Definition at line 81 of file gatherstats.py.

## 5.6 goal Namespace Reference

### Classes

- class [CircleGoal](#)  
*Object that holds data about a goal modelled as circular configuration space.*

### Variables

- string [\\_\\_author\\_\\_](#) = "Alex Wallar <aw204@st-andrews.ac.uk>"

### 5.6.1 Variable Documentation

5.6.1.1 `string goal.__author__ = "Alex Wallar <aw204@st-andrews.ac.uk>"`

Definition at line 3 of file goal.py.

## 5.7 mapparser Namespace Reference

### Functions

- def [mapVal](#)  
*Maps a value that is between `in_min` and `in_max` to a value between `out_min` and `out_max`.*
- def [mparse](#)  
*Parses a map file into a list of obstacles.*

### 5.7.1 Function Documentation

5.7.1.1 `def mapparser.mapVal( x, in_min, in_max, out_min, out_max )`

Maps a value that is between `in_min` and `in_max` to a value between `out_min` and `out_max`.

#### Parameters

<code>in_min</code>	The minimum value that the input value could be
<code>in_max</code>	The maximum value that the input value could be
<code>out_min</code>	The minimum value that the output value could be
<code>out_max</code>	The maximum value that the output value could be

#### Returns

A scaled value based on a given input

Definition at line 16 of file mapparser.py.

Here is the caller graph for this function:



5.7.1.2 `def mapparser.mparse( filename, staticObstacleList = list(), kwargs )`

Parses a map file into a list of obstacles.



## Parameters

<i>filename</i>	The file name of the map file
-----------------	-------------------------------

## Returns

A list of obstacles

Definition at line 31 of file mapparser.py.

Here is the call graph for this function:



## 5.8 obstacle Namespace Reference

## Classes

- class [PolyObstacle](#)

*Object that represents the an obstacle represented by a series of points (in the node list) which make up a set of lines.*

## Variables

- string `__author__` = "Alex Wallar <aw204@st-andrews.ac.uk>"

### 5.8.1 Variable Documentation

5.8.1.1 string `obstacle.__author__` = "Alex Wallar <aw204@st-andrews.ac.uk>"

Definition at line 4 of file obstacle.py.

## 5.9 priodict Namespace Reference

## Classes

- class [priorityDictionary](#)

## 5.10 prm Namespace Reference

## Classes

- class [PRMGenerator](#)

*Class used to hold methods and variables that are important for the global path planning problem.*

## 5.11 test\_sim Namespace Reference

### Variables

- dictionary `mapDict`  
*Main module to run for testing purposes.*
- tuple `flockSize` = `int(sys.argv[2])`
- list `startPoint` = `mapDict[sys.argv[1]]`
- list `endPoint` = `mapDict[sys.argv[1]]`
- list `mapFilePath` = `sys.argv[1]`
- `obstacleFilePath` = `None`
- `dynamicObstacleAutoGenerate` = `False`
- int `generateTarget` = `0`
- list `arg` = `sys.argv[3]`
- tuple `fs`

### 5.11.1 Variable Documentation

#### 5.11.1.1 list `test_sim.arg` = `sys.argv[3]`

Definition at line 54 of file `test_sim.py`.

#### 5.11.1.2 `test_sim.dynamicObstacleAutoGenerate` = `False`

Definition at line 50 of file `test_sim.py`.

#### 5.11.1.3 list `test_sim.endPoint` = `mapDict[sys.argv[1]]`

Definition at line 47 of file `test_sim.py`.

#### 5.11.1.4 tuple `test_sim.flockSize` = `int(sys.argv[2])`

Definition at line 45 of file `test_sim.py`.

#### 5.11.1.5 tuple `test_sim.fs`

**Initial value:**

```
1 = bs.FlockSim(
2     flockSize,
3     startPoint,
4     endPoint,
5     map_file=mapFilePath,
6     obstacle_file=obstacleFilePath,
7     auto_gen_obst=dynamicObstacleAutoGenerate,
8     auto_gen_number=generateTarget
9 )
```

Definition at line 61 of file `test_sim.py`.

#### 5.11.1.6 tuple `test_sim.generateTarget` = `0`

Definition at line 51 of file `test_sim.py`.

## 5.11.1.7 dictionary test\_sim.mapDict

Initial value:

```

1 = {
2     "maps/scene2.map": {
3         "startPoint": (494, 213),
4         "endPoint": (404, 20)
5     },
6     "maps/scene3.map": {
7         "startPoint": (356, 42),
8         "endPoint": (852, 450)
9     },
10    "maps/scenel.map": {
11        "startPoint": (50, 50), # (50, 600)
12        "endPoint": (980, 30)
13    },
14    "maps/empty.map": {
15        "startPoint": (50, 50), # (50, 600)
16        "endPoint": (980, 590)
17    },
18    "maps/s.map": {
19        "startPoint": (80, 80), # (50, 600)
20        "endPoint": (980, 30)
21    },
22    "maps/maze.map": {
23        "startPoint": (50, 50), # (50, 600)
24        "endPoint": (950, 30)
25    },
26    "maps/maze2.map": {
27        "startPoint": (50, 70), # (50, 600)
28        "endPoint": (950, 30)
29    }
30 }
```

Main module to run for testing purposes.

Definition at line 10 of file test\_sim.py.

## 5.11.1.8 list test\_sim.mapFilePath = sys.argv[1]

Definition at line 48 of file test\_sim.py.

## 5.11.1.9 test\_sim.obstacleFilePath = None

Definition at line 49 of file test\_sim.py.

## 5.11.1.10 list test\_sim.startPoint = mapDict[sys.argv[1]]

Definition at line 46 of file test\_sim.py.



## Chapter 6

# Class Documentation

### 6.1 boid.Boid Class Reference

Class which represents one boid.

#### Public Member Functions

- def `__init__`  
*Initializes all of the variables given as input to the constructor used by the boid.*
- def `sumDivide`  
*Special sort of reduce that sums components in a list of vectors and divides each final component with a certain number.*
- def `norm`  
*Gets the distance between two points.*
- def `getVar`  
*Gets multiple variables from a list with one call.*
- def `obstacleFunc`  
*Defines the potential between a boid and an obstacle.*
- def `mag`  
*Gets the magnitude of a vector.*
- def `sigmoidFunc`  
*Defines a sigmoidal curve used for goal attraction and for boid repulsion.*
- def `inGoal`  
*Checks if a piont is in the current goal.*
- def `inWorld`  
*Checks if a point is in the world.*
- def `pointAllowed`  
*Checks if a point is inside or collides with any of the obstacles.*
- def `initFunctionParameters`
- def `setBoidList`  
*Setter method used to set the list of boids.*
- def `updatePositionBuffer`  
*Updates the position buffer.*
- def `findMax`  
*Gets the n maximum values from a list.*
- def `reduceWeightValues`  
*Works in cohesion with sumDivide.*

- def [getDirectionVector](#)  
*Gets a scaled direction vector from an unscaled vector.*
- def [getObstacleVectorList](#)  
*Gets the potential vectors to a boid due to the repulsive obstacle field.*
- def [getGoalVector](#)  
*Gets the potential vectors to a boid due to the attractive goal field.*
- def [getBoidVectorList](#)  
*Gets the potential vectors to a boid due to the repulsive boid field.*
- def [getNeighborVectorList](#)  
*Gets the heading vectors of the neighbours.*
- def [setNewGoal](#)  
*Sets the new goal.*
- def [determineRandomWalk](#)  
*Increments the random walk counter, changes the current goal if necessary.*
- def [determineNewPath](#)  
*When the boid is stuck, it reweights the roadmap and finds a new suitable path.*
- def [update](#)  
*Updates the boid's heading and position due to the potential fields.*
- def [draw](#)  
*Draws the boid as a pygame circle in the pygame screen.*

## Public Attributes

- [gammaFunc](#)  
*Function used to choose a neighbour.*
- [prmGen](#)  
*Class which holds the details about the global path planner.*
- [screen](#)  
*PyGame screen.*
- [radius](#)  
*Radius of the boid.*
- [heading](#)  
*Initial random heading.*
- [dim](#)  
*Dimensions of the screen.*
- [ySize](#)
- [neighborSize](#)  
*Number of neighbours that will influence the boid.*
- [color](#)  
*Unique color used to distinguish the boid (only used in debugging and visualization)*
- [speed](#)  
*Maximum speed of the boid.*
- [obstacleList](#)  
*List of obstacles that were parsed by mapparser.*
- [goalList](#)  
*Goals used by the boid.*
- [goalCounter](#)  
*Used to store what goal the boid is currently looking at.*
- [goal](#)  
*Initializes the current goal.*

- [stuck](#)  
*Defines if the boid is stuck.*
- [sPos](#)  
*Starting position of the boid.*
- [ePos](#)  
*Position of the boids.*
- [position](#)  
*Sets the position of the boid.*
- [positionBuffer](#)  
*Used to tell if the boid is stuck or not.*
- [goalNodes](#)
- [roadmap](#)
- [endIndex](#)
- [obInfluenceR](#)  
*The radius of influence used when filtering the number of obstacles it needs to check.*
- [bInfluenceR](#)  
*The radius of influence used when filtering the number of boids it needs to check.*
- [obBeta](#)  
*Priori constant for obstacle repulsion (increasing it gives more priority to the repulsive obstacle field)*
- [gAlpha](#)  
*Scales the value returned by the sigmoid function for goal attraction.*
- [gBeta](#)  
*Helps scale the value returned by the sigmoid function for goal attraction.*
- [gDelta](#)  
*Constant that is used in the sigmoidal curve for goal attraction.*
- [gConst](#)  
*Priori constant for goal attraction (increasing it gives more priority to the attractive goal field)*
- [bAlpha](#)  
*Scales the value returned by the sigmoid function for boid repulsion.*
- [bBeta](#)  
*Helps scale the value returned by the sigmoid function for boid repulsion.*
- [bDelta](#)  
*Constant that is used in the sigmoid curve for boid repulsion.*
- [bConst](#)  
*Priori constant for boid repulsion (increasing it gives more priority to the repulsive boid field)*
- [stuckConst](#)  
*Amount of movement in the position buffer needs to be less than this value for a boid to be considered stuck.*
- [stuckDAvg](#)  
*The average distance from a neighbour when a boid is stuck.*
- [stuckDSigma](#)  
*The standard deviation for a neighbour probability distribution Helps boid pick closer neighbours when it is stuck.*
- [nStuckDAvg](#)  
*The average distance from a neighbour when a boid is not stuck.*
- [nStuckDSigma](#)  
*The standard deviation in a neighbour distance distribution when the boid is not stuck.*
- [randomWalkX](#)  
*Maximum x random walk.*
- [randomWalkY](#)  
*Maximum y random walk.*
- [randWalkCount](#)  
*Stores the number of times a random walk has occurred.*

- [headWeightList](#)

*Weights how much the previous heading affects the new heading.*

- [boidList](#)
- [compWeightList](#)

### 6.1.1 Detailed Description

Class which represents one boid.

Definition at line 30 of file boid.py.

### 6.1.2 Constructor & Destructor Documentation

6.1.2.1 `def boid.Boid.__init__( self, _sPos, _ePos, _speed, _xSize, _ySize, _neighborSize, _gammaFunc, _obstacleList, _goalList, _prmGen, _screen, _color )`

Initializes all of the variables given as input to the constructor used by the boid.

#### Parameters

<code>self</code>	The object pointer
<code>_sPos</code>	The starting position of the boid (some noise added when initializing the flock)
<code>_ePos</code>	The ending position of the flock
<code>_xSize</code>	The size of the x axis of the pygame screen
<code>_neighbourSize</code>	The number of neighbours that will influence the boid's heading
<code>_obstacleList</code>	List of obstacles generated by mapparser
<code>_goalList</code>	List of goals used by the boid
<code>_prmGen</code>	Object that stores all of the data about the global planner
<code>_screen</code>	PyGame screen
<code>_color</code>	Unique color used for debugging purposes

#### Returns

An instance of a boid

Definition at line 55 of file boid.py.

### 6.1.3 Member Function Documentation

6.1.3.1 `def boid.Boid.determineNewPath( self )`

When the boid is stuck, it reweights the roadmap and finds a new suitable path.

Definition at line 653 of file boid.py.

Here is the call graph for this function:

Here is the caller graph for this function:

6.1.3.2 `def boid.Boid.determineRandomWalk( self )`

Increments the random walk counter, changes the current goal if necessary.

#### Returns

New random walk vectors where the maximum component value is determined by the randomWalk fields

Definition at line 626 of file boid.py.



6.1.3.3 `def boid.Boid.draw ( self )`

Draws the boid as a pygame circle in the pygame screen.

Definition at line 768 of file boid.py.

6.1.3.4 `def boid.Boid.findMax ( self, searchThrough, counter )`

Gets the n maximum values from a list.

**Parameters**

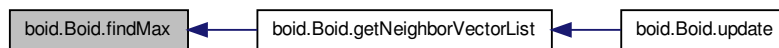
<i>searchThrough</i>	The list that the maximums will be extracted from
<i>counter</i>	The number of values to be extracted

**Returns**

A list of the counter maximum values from searchThrough

Definition at line 392 of file boid.py.

Here is the caller graph for this function:

6.1.3.5 `def boid.Boid.getBoidVectorList ( self )`

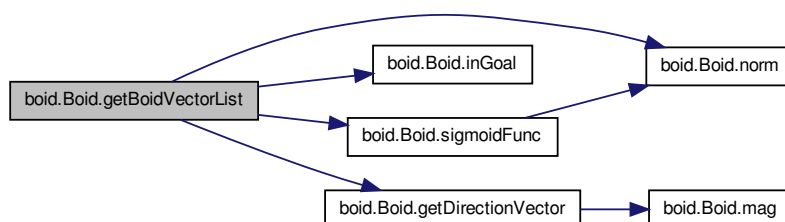
Gets the potential vectors to a boid due to the repulsive boid field.

**Returns**

A list of scaled vectors that will be used to determine the influence of the boids on the heading. Also returns the sum of the potential

Definition at line 521 of file boid.py.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 6.1.3.6 `def boid.Boid.getDirectionVector ( self, vector )`

Gets a scaled direction vector from an unscaled vector.

##### Parameters

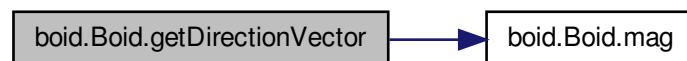
<i>vector</i>	Vector to be scaled
---------------	---------------------

##### Returns

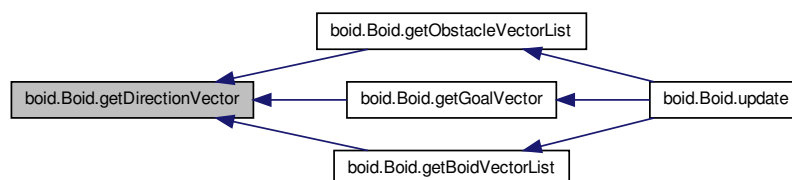
A vector whose maximum magnitude is less than the specified maximum speed

Definition at line 425 of file `boid.py`.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 6.1.3.7 `def boid.Boid.getGoalVector ( self )`

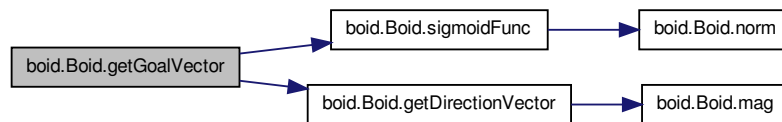
Gets the potential vectors to a boid due to the attractive goal field.

### Returns

A list of scaled vectors that will be used to determine the influence of the goal on the heading. Also returns the sum of the potential values

Definition at line 490 of file boid.py.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 6.1.3.8 def boid.Boid.getNeighborVectorList ( self )

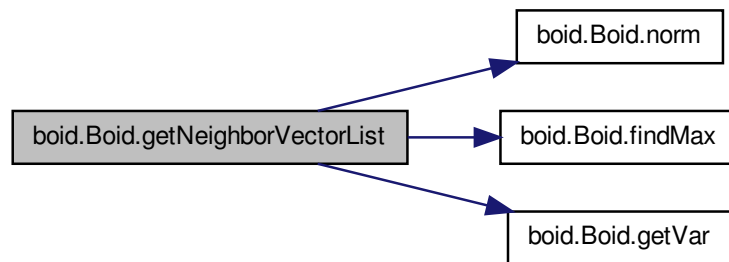
Gets the heading vectors of the neighbours.

**Returns**

A list of scaled vectors that represent the neighbour headings. Also returns the indicies in the boid list in which the neighbours are stored

Definition at line 576 of file boid.py.

Here is the call graph for this function:



Here is the caller graph for this function:



### 6.1.3.9 def boid.Boid.getObstacleVectorList ( self )

Gets the potential vectors to a boid due to the repulsive obstacle field.

**Returns**

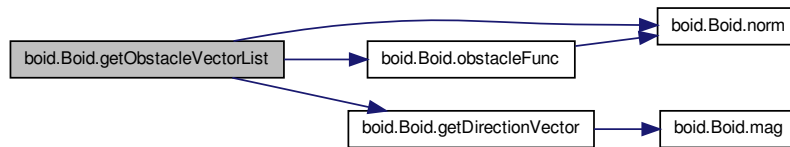
A list of scaled vectors that will be used to determine the influence of obstacles on the heading. Also returns the sum of the potential values

```

for ob in self.obstacleList:
    pygame.draw.circle(
self.screen,
(255,0,255),
map(int, ob.getPoint(self.position)),
2
    )
  
```

Definition at line 438 of file boid.py.

Here is the call graph for this function:



Here is the caller graph for this function:



6.1.3.10 `def boid.Boid.getVar ( self, searchList, ind )`

Gets multiple variables from a list with one call.

Parameters

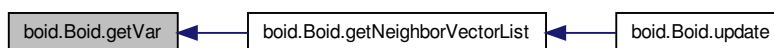
<i>searchList</i>	The list that the values will be taken from
<i>ind</i>	the indicies that will be queried

Returns

A list of values from search list

Definition at line 189 of file `boid.py`.

Here is the caller graph for this function:



6.1.3.11 `def boid.Boid.inGoal ( self, p )`

Checks if a piont is in the current goal.

**Parameters**

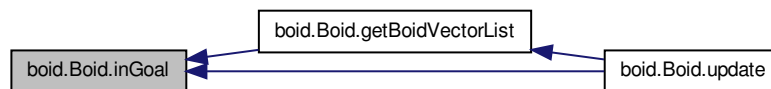
$p$	The point that is going to be checked
-----	---------------------------------------

**Returns**

A boolean value representing if the point is in the current goal

Definition at line 250 of file boid.py.

Here is the caller graph for this function:

**6.1.3.12 def boid.Boid.initFunctionParameters ( self )**

Definition at line 286 of file boid.py.

**6.1.3.13 def boid.Boid.inWorld ( self, p )**

Checks if a point is in the world.

**Parameters**

$p$	The point that is going to be checked
-----	---------------------------------------

**Returns**

A boolean value representing if the point is in the world

Definition at line 265 of file boid.py.

**6.1.3.14 def boid.Boid.mag ( self, vec )**

Gets the magnitude of a vector.

**Parameters**

$vec$	A vector represented as a list
-------	--------------------------------

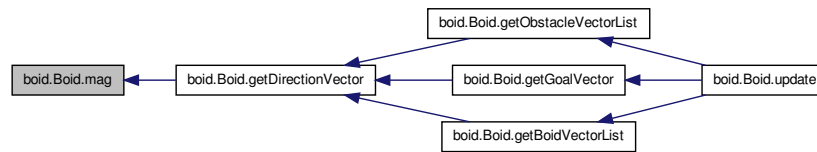
**Returns**

The magnitude of the vector

Definition at line 218 of file boid.py.

Here is the call graph for this function:

Here is the caller graph for this function:



#### 6.1.3.15 def boid.Boid.norm ( self, p1, p2 )

Gets the distance between two points.

Parameters

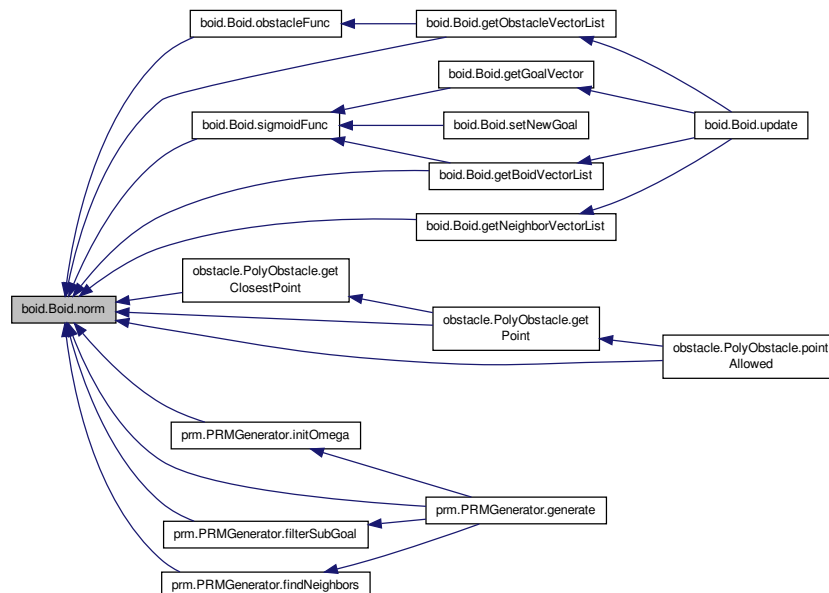
<i>p1,p2</i>	points whose distance will be returned
--------------	--

Returns

The Euclidean distance between p1 and p2

Definition at line 170 of file boid.py.

Here is the caller graph for this function:



#### 6.1.3.16 def boid.Boid.obstacleFunc ( self, beta, b, o )

Defines the potential between a boid and an obstacle.

## Parameters

<i>beta</i>	Constant used to increase the weight of the function
<i>b</i>	The boid that is being compared
<i>o</i>	The obstacle that is being compared

## Returns

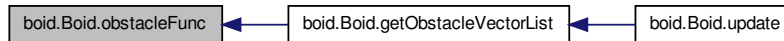
A value representing the potential between b and o

Definition at line 200 of file boid.py.

Here is the call graph for this function:



Here is the caller graph for this function:



### 6.1.3.17 def boid.Boid.pointAllowed ( self, p )

Checks if a point is inside or collides with any of the obstacles.

## Parameters

<i>p</i>	The point that will be checked
----------	--------------------------------

Definition at line 278 of file boid.py.

### 6.1.3.18 def boid.Boid.reduceWeightValues ( self, wList, vList )

Works in cohesion with sumDivide.

Weights values in a list and divides by the sum of those weights

## Parameters

<i>wList</i>	List of Weights
<i>*vList</i>	Values that will be weighted

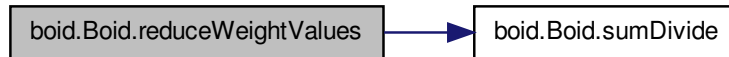


**Returns**

An average vector that represents the average heading due to the potential fields

Definition at line 412 of file boid.py.

Here is the call graph for this function:

**6.1.3.19** `def boid.Boid.setBoidList ( self, _boidList )`

Setter method used to set the list of boids.

**Parameters**

<code>_boidList</code>	The list of boids in the flock
------------------------	--------------------------------

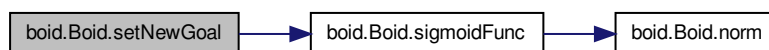
Definition at line 370 of file boid.py.

**6.1.3.20** `def boid.Boid.setNewGoal ( self )`

Sets the new goal.

Definition at line 606 of file boid.py.

Here is the call graph for this function:

**6.1.3.21** `def boid.Boid.sigmoidFunc ( self, alpha, beta, delta, const, b_r, g_r, b_pos, g_pos )`

Defines a sigmoidal curve used for goal attraction and for boid repulsion.

**Parameters**

<code>alpha, beta, delta, const</code>	Constants that are used to modify the shape of the curve
<code>b_r, b_pos</code>	The radius and position of the boid

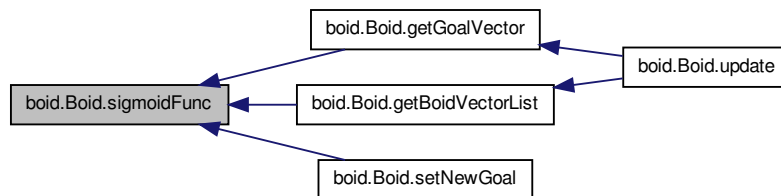
<code>g_r,g_pos</code>	The radius and position of a goal / boid
------------------------	--

Definition at line 231 of file boid.py.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 6.1.3.22 `def boid.Boid.sumDivide ( self, lt, s )`

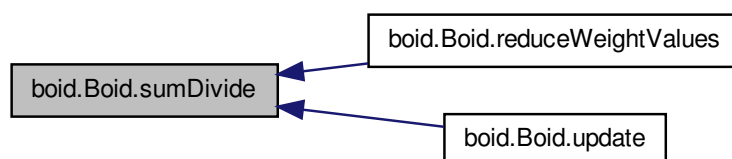
Special sort of reduce that sums components in a list of vectors and divides each final component with a certain number.

##### Parameters

<code>self</code>	The object pointer
<code>lt</code>	List of vectors that will be summed over and divided
<code>s</code>	Number that will divide each component by at the end

Definition at line 149 of file boid.py.

Here is the caller graph for this function:

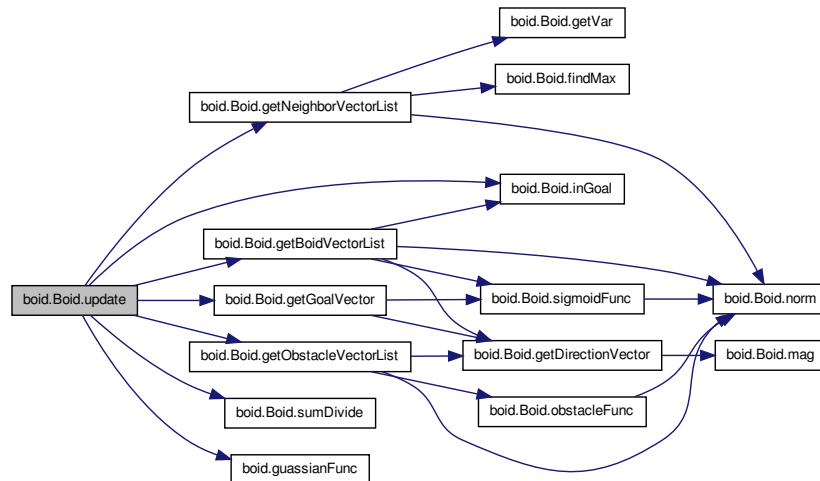


#### 6.1.3.23 def boid.Boid.update ( self )

Updates the boid's heading and position due to the potential fields.

Definition at line 690 of file boid.py.

Here is the call graph for this function:



#### 6.1.3.24 def boid.Boid.updatePositionBuffer ( self )

Updates the position buffer.

##### Returns

The displacement of a boid over a certain number of frames

Definition at line 378 of file boid.py.

### 6.1.4 Member Data Documentation

#### 6.1.4.1 boid.Boid.bAlpha

Scales the value returned by the sigmoid function for boid repulsion.

Definition at line 319 of file boid.py.

#### 6.1.4.2 boid.Boid.bBeta

Helps scale the value returned by the sigmoid function for boid repulsion.

Definition at line 323 of file boid.py.

#### 6.1.4.3 boid.Boid.bConst

Priroi constant for boid repulsion (increasing it gives more priority to the repulsive boid field)

Definition at line 331 of file boid.py.

#### 6.1.4.4 `boid.Boid.bDelta`

Constant that is used in the sigmoid curve for boid repulsion.

Definition at line 327 of file `boid.py`.

#### 6.1.4.5 `boid.Boid.bInfluenceR`

The radius of influence used when filtering the number of boids it needs to check.

Definition at line 295 of file `boid.py`.

#### 6.1.4.6 `boid.Boid.boidList`

Definition at line 371 of file `boid.py`.

#### 6.1.4.7 `boid.Boid.color`

Unique color used to distinguish the boid (only used in debugging and visualization)

Definition at line 87 of file `boid.py`.

#### 6.1.4.8 `boid.Boid.compWeightList`

Definition at line 731 of file `boid.py`.

#### 6.1.4.9 `boid.Boid.dim`

Dimensions of the screen.

Definition at line 80 of file `boid.py`.

#### 6.1.4.10 `boid.Boid.endIndex`

Definition at line 136 of file `boid.py`.

#### 6.1.4.11 `boid.Boid.ePos`

Position of the boids.

Definition at line 111 of file `boid.py`.

#### 6.1.4.12 `boid.Boid.gAlpha`

Scales the value returned by the sigmoid function for goal attraction.

Definition at line 303 of file `boid.py`.

#### 6.1.4.13 `boid.Boid.gammaFunc`

Function used to choose a neighbour.

Definition at line 58 of file `boid.py`.

#### 6.1.4.14 boid.Boid.gBeta

Helps scale the value returned by the sigmoid function for goal attraction.

Definition at line 307 of file boid.py.

#### 6.1.4.15 boid.Boid.gConst

Priori constant for goal attraction (increasing it gives more priority to the attractive goal field)

Definition at line 315 of file boid.py.

#### 6.1.4.16 boid.Boid.gDelta

Constant that is used in the sigmoidal curve for goal attraction.

Definition at line 311 of file boid.py.

#### 6.1.4.17 boid.Boid.goal

Initializes the current goal.

Definition at line 102 of file boid.py.

#### 6.1.4.18 boid.Boid.goalCounter

Used to store what goal the boid is currently looking at.

Definition at line 99 of file boid.py.

#### 6.1.4.19 boid.Boid.goalList

Goals used by the boid.

Definition at line 96 of file boid.py.

#### 6.1.4.20 boid.Boid.goalNodes

Definition at line 132 of file boid.py.

#### 6.1.4.21 boid.Boid.heading

Initial random heading.

Definition at line 70 of file boid.py.

#### 6.1.4.22 boid.Boid.headWeightList

Weights how much the previous heading affects the new heading.

Definition at line 363 of file boid.py.

#### 6.1.4.23 `boid.Boid.neighborSize`

Number of neighbours that will influence the boid.

Definition at line 83 of file `boid.py`.

#### 6.1.4.24 `boid.Boid.nStuckDAvg`

The average distance from a neighbour when a boid is not stuck.

Definition at line 347 of file `boid.py`.

#### 6.1.4.25 `boid.Boid.nStuckDSigma`

The standard deviation in a neighbour distance distribution when the boid is not stuck.

Definition at line 351 of file `boid.py`.

#### 6.1.4.26 `boid.Boid.obBeta`

Priori constant for obstacle repulsion (increasing it gives more priority to the repulsive obstacle field)

Definition at line 299 of file `boid.py`.

#### 6.1.4.27 `boid.Boid.obInfluenceR`

The radius of influence used when filtering the number of obstacles it needs to check.

Definition at line 291 of file `boid.py`.

#### 6.1.4.28 `boid.Boid.obstacleList`

List of obstacles that were parsed by mapparser.

Definition at line 93 of file `boid.py`.

#### 6.1.4.29 `boid.Boid.position`

Sets the position of the boid.

Definition at line 114 of file `boid.py`.

#### 6.1.4.30 `boid.Boid.positionBuffer`

Used to tell if the boid is stuck or not.

Definition at line 125 of file `boid.py`.

#### 6.1.4.31 `boid.Boid.prmGen`

Class which holds the details about the global path planner.

Definition at line 61 of file `boid.py`.

**6.1.4.32 boid.Boid.radius**

Radius of the boid.

Definition at line 67 of file boid.py.

**6.1.4.33 boid.Boid.randomWalkX**

Maximum x random walk.

Definition at line 354 of file boid.py.

**6.1.4.34 boid.Boid.randomWalkY**

Maximum y random walk.

Definition at line 357 of file boid.py.

**6.1.4.35 boid.Boid.randWalkCount**

Stores the number of times a random walk has occurred.

Definition at line 360 of file boid.py.

**6.1.4.36 boid.Boid.roadmap**

Definition at line 135 of file boid.py.

**6.1.4.37 boid.Boid.screen**

PyGame screen.

Definition at line 64 of file boid.py.

**6.1.4.38 boid.Boid.speed**

Maximum speed of the boid.

Definition at line 90 of file boid.py.

**6.1.4.39 boid.Boid.sPos**

Starting position of the boid.

Definition at line 108 of file boid.py.

**6.1.4.40 boid.Boid.stuck**

Defines if the boid is stuck.

Definition at line 105 of file boid.py.

#### 6.1.4.41 `boid.Boid.stuckConst`

Amount of movement in the position buffer needs to be less than this value for a boid to be considered stuck.

Definition at line 335 of file `boid.py`.

#### 6.1.4.42 `boid.Boid.stuckDAvg`

The average distance from a neighbour when a boid is stuck.

Used to pick closer neighbours when stuck to help get out of the situation

Definition at line 340 of file `boid.py`.

#### 6.1.4.43 `boid.Boid.stuckDSigma`

The standard deviation for a neighbour probability distribution Helps boid pick closer neighbours when it is stuck.

Definition at line 344 of file `boid.py`.

#### 6.1.4.44 `boid.Boid.ySize`

Definition at line 80 of file `boid.py`.

The documentation for this class was generated from the following file:

- [code/boid.py](#)

## 6.2 `goal.CircleGoal` Class Reference

Object that holds data about a goal modelled as circular configuration space.

### Public Member Functions

- `def __init__`  
*Creates an instance of the [CircleGoal](#).*
- `def draw`  
*Draws the circle onto the pygame screen.*

### Public Attributes

- `screen`  
*PyGame screen used to draw the circle goal.*
- `colors`  
*List of colors given by PyGame.*
- `radius`  
*The radius of the circle goal.*
- `position`  
*The position of the circle goal.*



### 6.2.1 Detailed Description

Object that holds data about a goal modelled as circular configuration space.

Definition at line 12 of file goal.py.

### 6.2.2 Constructor & Destructor Documentation

#### 6.2.2.1 `def goal.CircleGoal.__init__( self, _radius, _position, _screen )`

Creates an instance of the [CircleGoal](#).

Parameters

<code>_radius</code>	The radius of the circle goal
<code>_position</code>	The position of the circle goal
<code>_screen</code>	The pygame screen used to draw the circle

Definition at line 20 of file goal.py.

### 6.2.3 Member Function Documentation

#### 6.2.3.1 `def goal.CircleGoal.draw( self )`

Draws the circle onto the pygame screen.

Definition at line 38 of file goal.py.

### 6.2.4 Member Data Documentation

#### 6.2.4.1 `goal.CircleGoal.colors`

List of colors given by PyGame.

Definition at line 26 of file goal.py.

#### 6.2.4.2 `goal.CircleGoal.position`

The position of the circle goal.

Definition at line 32 of file goal.py.

#### 6.2.4.3 `goal.CircleGoal.radius`

The radius of the circle goal.

Definition at line 29 of file goal.py.

#### 6.2.4.4 `goal.CircleGoal.screen`

PyGame screen used to draw the circle goal.

Definition at line 23 of file goal.py.

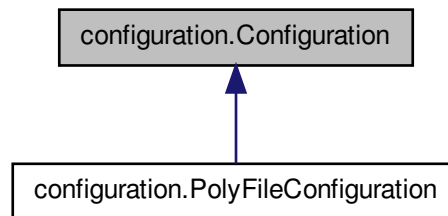
The documentation for this class was generated from the following file:

- [code/goal.py](#)

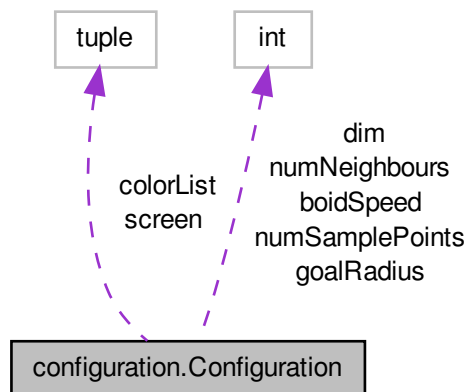
### 6.3 configuration.Configuration Class Reference

Static class that holds important global variables.

Inheritance diagram for configuration.Configuration:



Collaboration diagram for configuration.Configuration:



#### Static Public Attributes

- `int dim = 1000`  
*Dimensions of the screen.*
- `int numSamplePoints = 300`  
*Number of sample points to use in the PRM.*
- `int goalRadius = 20`  
*Defines the radius of all goals.*
- `int boidSpeed = 30`  
*Maximum speed of the boids.*
- `int numNeighbours = 3`  
*Number of neighbours the boids will influence a boid's heading.*

- tuple `screen` = `pygame.display.set_mode(dim)`  
*The screen used to draw the simulation.*
- tuple `colorList`  
*The list of colors (used for debugging purposes)*

### 6.3.1 Detailed Description

Static class that holds important global variables.

Definition at line 16 of file `configuration.py`.

### 6.3.2 Member Data Documentation

#### 6.3.2.1 `int configuration.Configuration.boidSpeed = 30` `[static]`

Maximum speed of the boids.

Definition at line 28 of file `configuration.py`.

#### 6.3.2.2 `tuple configuration.Configuration.colorList` `[static]`

**Initial value:**

```
1 = map(
2     lambda k: color.THECOLORS[k],
3     color.THECOLORS.keys()
4 )
```

The list of colors (used for debugging purposes)

Definition at line 38 of file `configuration.py`.

#### 6.3.2.3 `int configuration.Configuration.dim = 1000` `[static]`

Dimensions of the screen.

Definition at line 19 of file `configuration.py`.

#### 6.3.2.4 `int configuration.Configuration.goalRadius = 20` `[static]`

Defines the radius of all goals.

Definition at line 25 of file `configuration.py`.

#### 6.3.2.5 `int configuration.Configuration.numNeighbours = 3` `[static]`

Number of neighbours the boids will influence a boid's heading.

Definition at line 32 of file `configuration.py`.

#### 6.3.2.6 `int configuration.Configuration.numSamplePoints = 300` `[static]`

Number of sample points to use in the PRM.

Definition at line 22 of file `configuration.py`.

### 6.3.2.7 tuple `configuration.Configuration.screen = pygame.display.set_mode(dim)` [static]

The screen used to draw the simulation.

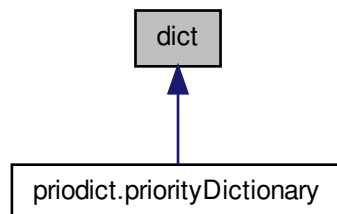
Definition at line 35 of file `configuration.py`.

The documentation for this class was generated from the following file:

- [code/configuration.py](#)

## 6.4 dict Class Reference

Inheritance diagram for dict:



The documentation for this class was generated from the following file:

- [code/priodict.py](#)

## 6.5 boidsimulation.FlockSim Class Reference

Main class for that is used for the simulation and display of the flock.

### Public Member Functions

- `def __init__`  
*Initializes the flock and the display mechanism (PyGame)*
- `def avg`  
*Gets the average of a list.*
- `def getStats`  
*Gets runtime statistics about the simulation and writes it to a file.*
- `def getBoidData`  
*Writes all of the boid positions to a file.*
- `def animate`  
*Renders and then allows interactive playback of the swarm simulation data.*
- `def init_prm`  
*Initializes the PRM generator used for the global planner.*
- `def render`

*Renders the scene.*

- def [play](#)

*Plays the scene after it has rendered.*

## Public Attributes

- [done](#)

*Tells if the flock has reached the end goal (used again to see if the escape or space bar were hit to stop the rendering)*

- [BLACK](#)

*Defines the color black.*

- [WHITE](#)

*Defines the color white.*

- [font](#)

*The font that is used for displaying the frame number.*

- [dim](#)

*The dimensions of the PyGame screen.*

- [config](#)

*The configuration object (in this case the configuration is defined by an exterior file)*

- [sPos](#)

*The starting point of the flock.*

- [ePos](#)

*The position of the last goal for the flock.*

- [surfaceList](#)

*Variable used to store the list of surfaces for simulation playback.*

- [iterations](#)

*Maximum number of iterations.*

- [frameCounter](#)

*Counts which frame the user is on for the playback (don't know why it is set to -2, it just works)*

- [counter](#)

*Global counter used for the rendering and the playback.*

- [flockSize](#)

*The size of the flock (number of boids)*

- [mapFile](#)

*The file that contains the data about the obstacles.*

- [dataFile](#)

*The file that the statistics data will be written to.*

- [obstacleFile](#)

*File containing the obstacles map points.*

- [auto\\_gen\\_obst](#)

*Auto Generate Random obstacles Flag.*

- [auto\\_gen\\_number](#)

*Number of random dynamic obstacles to generate.*

- [startTime](#)

- [numInGoal](#)

### 6.5.1 Detailed Description

Main class for that is used for the simulation and display of the flock.

It also is used to gather statistics about the flock and present them in a useful manner.

Definition at line 17 of file boidsimulation.py.

## 6.5.2 Constructor & Destructor Documentation

### 6.5.2.1 `def boidsimulation.FlockSim.__init__( self, flockSize, startPoint, endPoint, kwargs )`

Initializes the flock and the display mechanism (PyGame)

Parameters

<i>flockSize</i>	The size of the flock (number of boids)
<i>startPoint</i>	The macro starting position of the flock
<i>endPoint</i>	The last goal point for the flock
<i>_mapFile</i>	The file containing details about the obstacles
<i>_dataFile</i>	The file that the data will be exported to

Definition at line 33 of file boidsimulation.py.

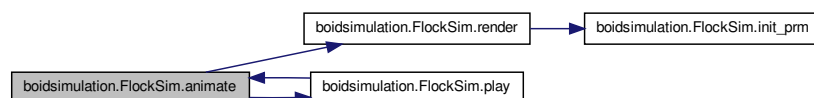
## 6.5.3 Member Function Documentation

### 6.5.3.1 `def boidsimulation.FlockSim.animate ( self )`

Renders and then allows interactive playback of the swarm simulation data.

Definition at line 167 of file boidsimulation.py.

Here is the call graph for this function:



Here is the caller graph for this function:



### 6.5.3.2 `def boidsimulation.FlockSim.avg ( self, l )`

Gets the average of a list.

Parameters

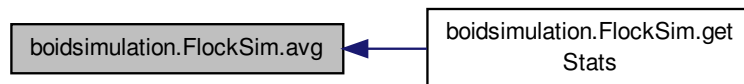
<i>l</i>	The list to be averaged
----------	-------------------------

**Returns**

l The average value in list l

Definition at line 99 of file boidsimulation.py.

Here is the caller graph for this function:

**6.5.3.3 def boidsimulation.FlockSim.getBoidData ( self )**

Writes all of the boid positions to a file.

Definition at line 143 of file boidsimulation.py.

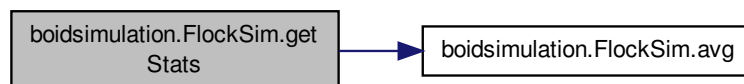
**6.5.3.4 def boidsimulation.FlockSim.getStats ( self )**

Gets runtime statistics about the simulation and writes it to a file.

Currently, the statistics being gathered are the current time that has passed, the average distance between the boids, the average minimum distance between the boids, and the number of boids that have finished

Definition at line 112 of file boidsimulation.py.

Here is the call graph for this function:

**6.5.3.5 def boidsimulation.FlockSim.init\_prm ( self )**

Initializes the PRM generator used for the global planner.

Also sets the boid list for the rest of the flock

Definition at line 176 of file boidsimulation.py.

Here is the caller graph for this function:



#### 6.5.3.6 def boidsimulation.FlockSim.play ( self )

Plays the scene after it has rendered.

Iterates through surfaces that have been stored in surfaceList and blits the new surface on the screen

Definition at line 264 of file boidsimulation.py.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 6.5.3.7 def boidsimulation.FlockSim.render ( self, forPlay = False )

Renders the scene.

This means that the time taken for the boids to reach the goal in this function is that actual amount of computational time needed.

**Parameters**

<i>forPlay</i>	Specifies if the surface data should be recorded for animation
----------------	--

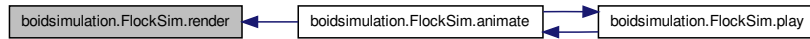
Definition at line 205 of file boidsimulation.py.

Here is the call graph for this function:





Here is the caller graph for this function:



## 6.5.4 Member Data Documentation

### 6.5.4.1 `boidsimulation.FlockSim.auto_gen_number`

Number of random dynamic obstacles to generate.

Definition at line 91 of file `boidsimulation.py`.

### 6.5.4.2 `boidsimulation.FlockSim.auto_gen_obst`

Auto Generate Random obstacles Flag.

Definition at line 88 of file `boidsimulation.py`.

### 6.5.4.3 `boidsimulation.FlockSim.BLACK`

Defines the color black.

Definition at line 41 of file `boidsimulation.py`.

### 6.5.4.4 `boidsimulation.FlockSim.config`

The configuration object (in this case the configuration is defined by an exterior file)

Definition at line 54 of file `boidsimulation.py`.

### 6.5.4.5 `boidsimulation.FlockSim.counter`

Global counter used for the rendering and the playback.

Definition at line 73 of file `boidsimulation.py`.

### 6.5.4.6 `boidsimulation.FlockSim.dataFile`

The file that the statistics data will be written to.

Definition at line 82 of file `boidsimulation.py`.

### 6.5.4.7 `boidsimulation.FlockSim.dim`

The dimensions of the PyGame screen.

Definition at line 50 of file `boidsimulation.py`.

#### 6.5.4.8 `boidsimulation.FlockSim.done`

Tells if the flock has reached the end goal (used again to see if the escape or space bar were hit to stop the rendering)

Definition at line 38 of file `boidsimulation.py`.

#### 6.5.4.9 `boidsimulation.FlockSim.ePos`

The position of the last goal for the flock.

Definition at line 60 of file `boidsimulation.py`.

#### 6.5.4.10 `boidsimulation.FlockSim.flockSize`

The size of the flock (number of boids)

Definition at line 76 of file `boidsimulation.py`.

#### 6.5.4.11 `boidsimulation.FlockSim.font`

The font that is used for displaying the frame number.

Definition at line 47 of file `boidsimulation.py`.

#### 6.5.4.12 `boidsimulation.FlockSim.frameCounter`

Counts which frame the user is on for the playback (don't know why it is set to -2, it just works)

Definition at line 70 of file `boidsimulation.py`.

#### 6.5.4.13 `boidsimulation.FlockSim.iterations`

Maximum number of iterations.

Definition at line 66 of file `boidsimulation.py`.

#### 6.5.4.14 `boidsimulation.FlockSim.mapFile`

The file that contains the data about the obstacles.

Definition at line 79 of file `boidsimulation.py`.

#### 6.5.4.15 `boidsimulation.FlockSim.numInGoal`

Definition at line 228 of file `boidsimulation.py`.

#### 6.5.4.16 `boidsimulation.FlockSim.obstacleFile`

File containing the obstacles map points.

Definition at line 85 of file `boidsimulation.py`.

#### 6.5.4.17 boidsimulation.FlockSim.sPos

The starting point of the flock.

Definition at line 57 of file boidsimulation.py.

#### 6.5.4.18 boidsimulation.FlockSim.startTime

Definition at line 209 of file boidsimulation.py.

#### 6.5.4.19 boidsimulation.FlockSim.surfaceList

Variable used to store the list of surfaces for simulation playback.

Definition at line 63 of file boidsimulation.py.

#### 6.5.4.20 boidsimulation.FlockSim.WHITE

Defines the color white.

Definition at line 44 of file boidsimulation.py.

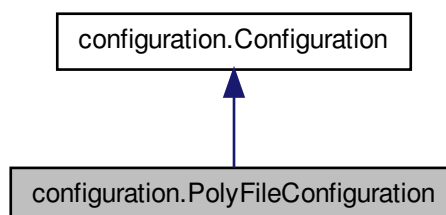
The documentation for this class was generated from the following file:

- [code/boidsimulation.py](#)

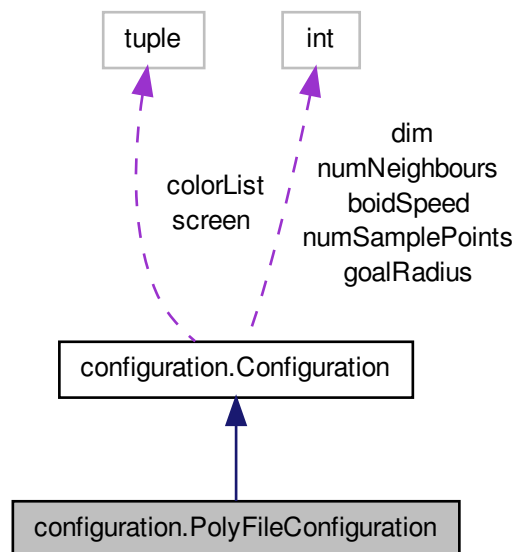
## 6.6 configuration.PolyFileConfiguration Class Reference

Extends the [Configuration](#) class.

Inheritance diagram for configuration.PolyFileConfiguration:



Collaboration diagram for configuration.PolyFileConfiguration:



## Public Member Functions

- def [parseDynamicObstacles](#)  
*Parses the obstacle map file and creates polygon objects with random behaviour by default.*
- def [autoGenerateDynamicObstacles](#)  
*Auto generate dynamic obstacles.*
- def [initVars](#)  
*Parses the file to get the obstacle list.*

## Public Attributes

- [nodes](#)
- [obstacleList](#)  
*List of obstacles parse static obstacles.*
- [auto\\_gen\\_obst](#)
- [auto\\_gen\\_number](#)
- [startPoint](#)  
*Starting point.*
- [endPoint](#)  
*Ending point.*
- [prmGen](#)  
*Object containing variables and mehtods for the global planner.*
- [goalList](#)  
*List of intermediate goals derived by the global planner.*
- [boidList](#)  
*List of boids in the flock.*

## Additional Inherited Members

### 6.6.1 Detailed Description

Extends the [Configuration](#) class.

This configuration gets the obstacles from .map files that have been created.

Definition at line 49 of file configuration.py.

### 6.6.2 Member Function Documentation

#### 6.6.2.1 `def configuration.PolyFileConfiguration.autoGenerateDynamicObstacles ( self )`

Auto generate dynamic obstacles.

Definition at line 105 of file configuration.py.

#### 6.6.2.2 `def configuration.PolyFileConfiguration.initVars ( self, startPoint, endPoint, flockSize, kwargs )`

Parses the file to get the obstacle list.

Creates a PRM generator to create a global map of the environment. Gets the list of intermediate goals. Also, creates the list of boids used in the simulation

Parameters

<i>startPoint</i>	The starting point for the boids
<i>endPoint</i>	The ending point for the boids
<i>flockSize</i>	The size of the flock (number of boids)
<i>filename</i>	The name of the file that contains the environment map

Definition at line 171 of file configuration.py.

#### 6.6.2.3 `def configuration.PolyFileConfiguration.parseDynamicObstacles ( self, dynamic_obstacles_fp )`

Parses the obstacle map file and creates polygon objects with random behaviour by default.

All obstacles (static/dynamic) obtains a list each other in the form of a list.

Definition at line 57 of file configuration.py.

### 6.6.3 Member Data Documentation

#### 6.6.3.1 `configuration.PolyFileConfiguration.auto_gen_number`

Definition at line 180 of file configuration.py.

#### 6.6.3.2 `configuration.PolyFileConfiguration.auto_gen_obst`

Definition at line 179 of file configuration.py.

#### 6.6.3.3 `configuration.PolyFileConfiguration.boidList`

List of boids in the flock.

Definition at line 205 of file configuration.py.

#### 6.6.3.4 configuration.PolyFileConfiguration.endPoint

Ending point.

Definition at line 188 of file configuration.py.

#### 6.6.3.5 configuration.PolyFileConfiguration.goalList

List of intermediate goals derived by the global planner.

Definition at line 202 of file configuration.py.

#### 6.6.3.6 configuration.PolyFileConfiguration.nodes

Definition at line 112 of file configuration.py.

#### 6.6.3.7 configuration.PolyFileConfiguration.obstacleList

List of obstacles parse static obstacles.

Definition at line 174 of file configuration.py.

#### 6.6.3.8 configuration.PolyFileConfiguration.prmGen

Object containing variables and methods for the global planner.

Definition at line 191 of file configuration.py.

#### 6.6.3.9 configuration.PolyFileConfiguration.startPoint

Starting point.

Definition at line 185 of file configuration.py.

The documentation for this class was generated from the following file:

- [code/configuration.py](#)

## 6.7 obstacle.PolyObstacle Class Reference

Object that represents the an obstacle represented by a series of points (in the node list) which make up a set of lines.

### Public Member Functions

- [def \\_\\_init\\_\\_](#)  
*Creates a [PolyObstacle](#) instance and initializes certain global variables.*
- [def removeSelfFromObstacleList](#)  
*Removes self from obstacle list.*
- [def norm](#)  
*Gets the Euclidean distance between p1 and p2.*
- [def estimatePoly](#)  
*Tries to estimate the polygon as a circle (very useful for environments with many obstacles i.e.*
- [def detectCollision](#)

- Detects a if there is a collision with the obstacle and the line <pStart, pEnd>*

  - def [getClosestPoint](#)

*Gets the closest point on line <a, b> to point p.*
  - def [rayintersectseg](#)

*Determines if a ray from point p intersects with an edge, edge.*
  - def [pointInPoly](#)

*Determines if a point p is inside the polygon represented by this [PolyObstacle](#) object.*
  - def [pointAllowed](#)

*Checks if a point is allowed, meaning no collisions occur.*
  - def [getPoint](#)

*Gets the closest point from the polygon to p.*
  - def [getRadius](#)

*Gets the 'radius' of the checking point.*
  - def [checkCollisionWithOtherObstacles](#)

*Check to see if there is a collision with a static obstacle.*
  - def [translate](#)

*Translate obstacle.*
  - def [determine\\_last\\_direction](#)
  - def [change\\_direction](#)

*Change direction.*
  - def [draw](#)

*Draws the polygon on the PyGame screen.*

## Public Attributes

- [nodes](#)

*A list of nodes used to represent the vertices.*
- [colors](#)

*A dictionary of colors defined in pygame.*
- [screen](#)

*The PyGame screen that is used to draw the obstacle.*
- [boundary](#)

*Bondaries of the simualation.*
- [dynamic](#)

*Defines wether the obstacle is dynamic or not.*
- [velocity](#)

*Velocity of the obstacle.*
- [displacement](#)

*The displacement of the obstacle.*
- [max\\_displacement](#)

*Max displacement allowed.*
- [obstacles](#)

*List of static obstacles.*
- [avgPoint](#)

*The average point in the polygon.*
- [maxDist](#)

*The maximum distance from any vertex and the average point.*

### 6.7.1 Detailed Description

Object that represents the an obstacle represented by a series of points (in the node list) which make up a set of lines.

These lines represent the exterior of an obstacle

Definition at line 18 of file obstacle.py.

### 6.7.2 Constructor & Destructor Documentation

#### 6.7.2.1 `def obstacle.PolyObstacle.__init__( self, _nodes, _screen, kwargs )`

Creates a [PolyObstacle](#) instance and initializes certain global variables.

Parameters

<code>_nodes</code>	A list of nodes used to represent the vertices of the polygon
<code>_screen</code>	The PyGame screen that is used to draw the obstacle

Definition at line 27 of file obstacle.py.

### 6.7.3 Member Function Documentation

#### 6.7.3.1 `def obstacle.PolyObstacle.change_direction( self, force_change=False, direction=None )`

Change direction.

Definition at line 434 of file obstacle.py.

Here is the call graph for this function:

Here is the caller graph for this function:

#### 6.7.3.2 `def obstacle.PolyObstacle.checkCollisionWithOtherObstacles( self, node )`

Check to see if there is a collision with a static obstacle.

Definition at line 359 of file obstacle.py.

Here is the call graph for this function:

Here is the caller graph for this function:

#### 6.7.3.3 `def obstacle.PolyObstacle.detectCollision( self, pStart, pEnd )`

Detects a if there is a collision with the obstacle and the line <pStart, pEnd>

Parameters

<code>pStart</code>	The starting point of the line
<code>pEnd</code>	The ending point of the line

Returns

A boolean value representing if a collision occurred

Definition at line 114 of file obstacle.py.



#### 6.7.3.4 def obstacle.PolyObstacle.determine\_last\_direction ( self )

Definition at line 418 of file obstacle.py.

Here is the caller graph for this function:

#### 6.7.3.5 def obstacle.PolyObstacle.draw ( self )

Draws the polygon on the PyGame screen.

Definition at line 478 of file obstacle.py.

Here is the call graph for this function:

#### 6.7.3.6 def obstacle.PolyObstacle.estimatePoly ( self )

Tries to estimate the polygon as a circle (very useful for environments with many obstacles i.e. a random field of obstacles)

Definition at line 81 of file obstacle.py.

#### 6.7.3.7 def obstacle.PolyObstacle.getClosestPoint ( self, a, b, p )

Gets the closest point on line <a, b> to point p.

##### Parameters

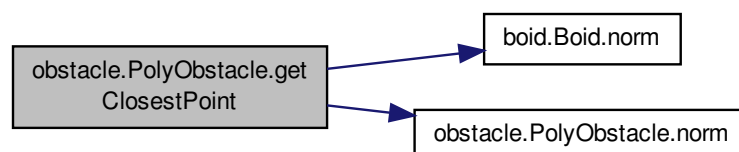
<i>a</i>	The starting point on the line
<i>b</i>	The ending point of the line
<i>p</i>	The point in which the closest distance will be checked

##### Returns

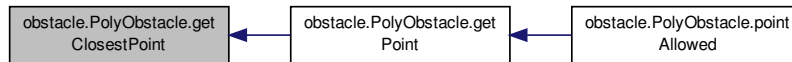
The closest point on line <a, b> to point p

Definition at line 171 of file obstacle.py.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 6.7.3.8 def obstacle.PolyObstacle.getPoint ( self, p )

Gets the closest point from the polygon to p.

##### Parameters

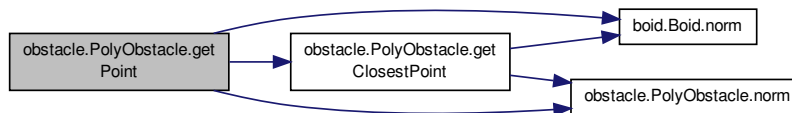
$p$	The point to be checked
-----	-------------------------

##### Returns

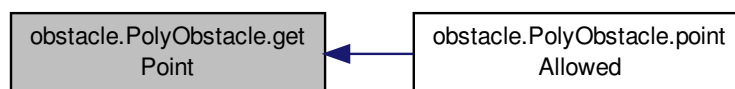
The closest point that lies on the polygon exterior to p

Definition at line 323 of file obstacle.py.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 6.7.3.9 def obstacle.PolyObstacle.getRadius ( self )

Gets the 'radius' of the checking point.

Only used for conformity with circle obstacles that have not been included in this repository

## Returns

1

Definition at line 352 of file obstacle.py.

### 6.7.3.10 def obstacle.PolyObstacle.norm ( self, p1, p2 )

Gets the Eulidean distance between p1 and p2.

## Parameters

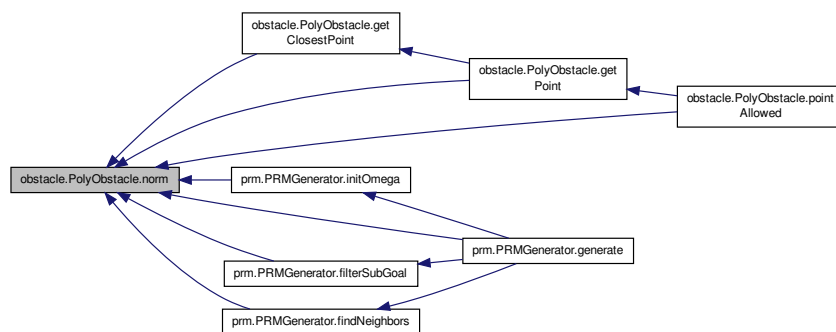
<i>p1,p2</i>	Points in space
--------------	-----------------

## Returns

The distance between p1 and p2

Definition at line 73 of file obstacle.py.

Here is the caller graph for this function:



### 6.7.3.11 def obstacle.PolyObstacle.pointAllowed ( self, b, p )

Checks if a point is allowed, meaning no collisions occur.

## Parameters

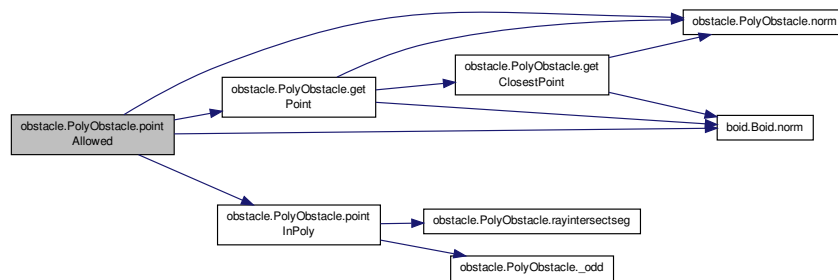
<i>b</i>	The boid object that will be checked
<i>p</i>	The point that will be checked

**Returns**

True if allowed, false otherwise

Definition at line 303 of file obstacle.py.

Here is the call graph for this function:



#### 6.7.3.12 def obstacle.PolyObstacle.pointInPoly ( self, p )

Determines if a point p is inside the polygon represented by this [PolyObstacle](#) object.

It does this by checking the number ray intersections that occur is odd or even. If the number is odd, the point is inside the polygon, otherwise it is not.

**Parameters**

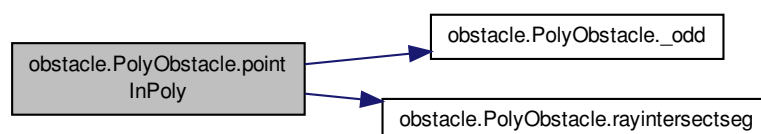
<i>p</i>	The point to be checked
----------	-------------------------

**Returns**

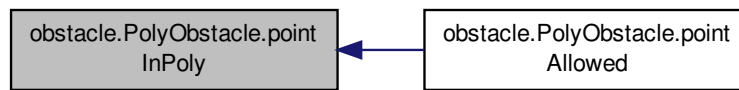
True if the point is in the polygon and false otherwise

Definition at line 286 of file obstacle.py.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 6.7.3.13 `def obstacle.PolyObstacle.rayintersectseg ( self, p, edge )`

Determines if a ray from point `p` intersects with an edge, `edge`.

Used to determine if a point `p` is inside the polygon

##### Parameters

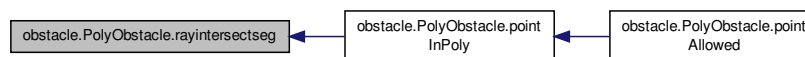
<i>p</i>	The point to be checked
<i>edge</i>	The edge that will be checked

##### Returns

True if a ray from point `p` intersects with edge and false otherwise

Definition at line 239 of file `obstacle.py`.

Here is the caller graph for this function:



#### 6.7.3.14 `def obstacle.PolyObstacle.removeSelfFromObstacleList ( self )`

Removes self from obstacle list.

Definition at line 62 of file `obstacle.py`.

#### 6.7.3.15 `def obstacle.PolyObstacle.translate ( self )`

Translate obstacle.

Definition at line 372 of file `obstacle.py`.

Here is the call graph for this function:

Here is the caller graph for this function:

## 6.7.4 Member Data Documentation

### 6.7.4.1 `obstacle.PolyObstacle.avgPoint`

The average point in the polygon.

Represents the center of the enclosing circle

Definition at line 85 of file `obstacle.py`.

### 6.7.4.2 `obstacle.PolyObstacle.boundary`

Bondaries of the simualation.

Definition at line 39 of file `obstacle.py`.

### 6.7.4.3 `obstacle.PolyObstacle.colors`

A dictionary of colors defined in pygame.

Definition at line 33 of file `obstacle.py`.

### 6.7.4.4 `obstacle.PolyObstacle.displacement`

The displacement of the obstacle.

Definition at line 48 of file `obstacle.py`.

### 6.7.4.5 `obstacle.PolyObstacle.dynamic`

Defines wether the obstacle is dynamic or not.

Definition at line 42 of file `obstacle.py`.

### 6.7.4.6 `obstacle.PolyObstacle.max_displacement`

Max displacement allowed.

Definition at line 51 of file `obstacle.py`.

### 6.7.4.7 `obstacle.PolyObstacle.maxDist`

The maximum distance from any vertex and the average point.

Definition at line 97 of file `obstacle.py`.

### 6.7.4.8 `obstacle.PolyObstacle.nodes`

A list of nodes used to represent the vertices.

Definition at line 30 of file `obstacle.py`.

### 6.7.4.9 `obstacle.PolyObstacle.obstacles`

List of static obstacles.

Definition at line 54 of file `obstacle.py`.

#### 6.7.4.10 obstacle.PolyObstacle.screen

The PyGame screen that is used to draw the obstacle.

Definition at line 36 of file obstacle.py.

#### 6.7.4.11 obstacle.PolyObstacle.velocity

Velocity of the obstacle.

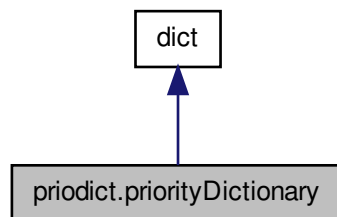
Definition at line 45 of file obstacle.py.

The documentation for this class was generated from the following file:

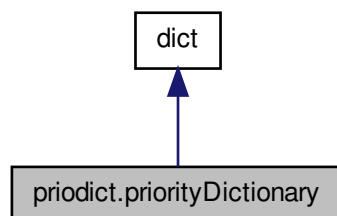
- code/[obstacle.py](#)

## 6.8 priodict.priorityDictionary Class Reference

Inheritance diagram for priodict.priorityDictionary:



Collaboration diagram for priodict.priorityDictionary:



### Public Member Functions

- def [\\_\\_init\\_\\_](#)

Initialize [priorityDictionary](#) by creating binary heap of pairs (value,key).

- def [smallest](#)

Find smallest item after removing deleted items from heap.

- def [\\_\\_iter\\_\\_](#)

Create destructive sorted iterator of [priorityDictionary](#).

- def [\\_\\_setitem\\_\\_](#)

Change value stored in dictionary and add corresponding pair to heap.

- def [setdefault](#)

Reimplement setdefault to call our customized **setitem**.

### 6.8.1 Detailed Description

Definition at line 6 of file priodict.py.

### 6.8.2 Constructor & Destructor Documentation

#### 6.8.2.1 def priodict.priorityDictionary.\_\_init\_\_( self )

Initialize [priorityDictionary](#) by creating binary heap of pairs (value,key).

Note that changing or removing a dict entry will not remove the old pair from the heap until it is found by [smallest\(\)](#) or until the heap is rebuilt.

Definition at line 12 of file priodict.py.

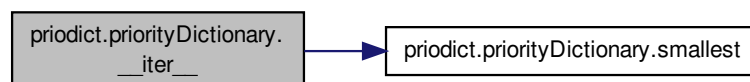
### 6.8.3 Member Function Documentation

#### 6.8.3.1 def priodict.priorityDictionary.\_\_iter\_\_( self )

Create destructive sorted iterator of [priorityDictionary](#).

Definition at line 39 of file priodict.py.

Here is the call graph for this function:



#### 6.8.3.2 def priodict.priorityDictionary.\_\_setitem\_\_( self, key, val )

Change value stored in dictionary and add corresponding pair to heap.

Rebuilds the heap if the number of deleted items grows too large, to avoid memory leakage.

Definition at line 51 of file priodict.py.



### 6.8.3.3 `def priodict.priorityDictionary.setdefault ( self, key, val )`

Reimplement setdefault to call our customized **setitem**.

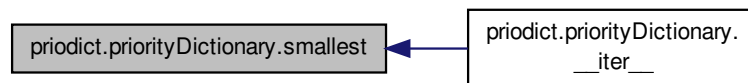
Definition at line 69 of file priodict.py.

### 6.8.3.4 `def priodict.priorityDictionary.smallest ( self )`

Find smallest item after removing deleted items from heap.

Definition at line 18 of file priodict.py.

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

- [code/priodict.py](#)

## 6.9 prm.PRMGenerator Class Reference

Class used to hold methods and variables that are important for the global path planning problem.

### Public Member Functions

- `def \_\_init\_\_`  
*Creates a new instance of the [PRMGenerator](#).*
- `def norm`  
*Gets the distance between p1 and p2.*
- `def generatePositionList`  
*Generates the random positions for the sample points.*
- `def initOmega`  
*Initiates the omega function which holds the node weights.*
- `def filterSubGoal`  
*Filters out sample points that are inside of obstacles or otherwise inadequate.*
- `def findNeighbors`  
*Finds suitable neighbours for a sample point.*
- `def getRandom`  
*Gets a random number and catches the ValueError if the two numbers are the same.*
- `def generate`  
*Generates a series of random points that will become the roadmap and connects them and weights them into a graph.*
- `def getShortestPath`
- `def draw`  
*Draws the graph.*
- `def drawPath`  
*Draws the selected shortest path.*

## Public Attributes

- [obstacleList](#)  
*List of obstacles.*
- [startPos](#)  
*Position of the first goal.*
- [endPos](#)  
*Position of the last goal.*
- [screen](#)  
*PyGame screen that is will be drawn on.*
- [xSize](#)  
*Horizontal size of the PyGame screen.*
- [ySize](#)  
*Vertical size of the PyGame screen.*
- [adjacentThresh](#)  
*Distance that the PRM is willing to check when connecting sample points.*
- [numNext](#)  
*Maximum number of sample points that can be connected.*
- [subGoalNumber](#)  
*Number of initial sample points.*
- [subGoalPositionList](#)  
*Initial positions of the sample points.*
- [roadmap](#)  
*The global roadmap.*
- [gPosList](#)  
*Holds the positions of the intermediate goals that were selected by the global path planner.*
- [goalNodes](#)  
*Indexes of the goal positions.*
- [omegaDict](#)  
*Dictionary (for easy access) that holds the weights for the nodes.*
- [dontDraw](#)

### 6.9.1 Detailed Description

Class used to hold methods and variables that are important for the global path planning problem.

This class generates the roadmap and finds the shortest path to the the goals by determining intermediate goals for the boids to be attracted to

Definition at line 16 of file prm.py.

### 6.9.2 Constructor & Destructor Documentation

6.9.2.1 `def prm.PRMGenerator.__init__( self, _startPos, _endPos, _obstacleList, _xSize, _ySize, _subGoalNumber, _screen )`

Creates a new instance of the [PRMGenerator](#).

Intializes key variables used in the generation of the global planner.

## Parameters

<code>_startPos</code>	The starting position of the boids
<code>_endPos</code>	The final goal position of the boids
<code>_obstacleList</code>	The list of obstacles that the global planner needs to avoid
<code>_xSize</code>	The size of the x component of the screen
<code>_ySize</code>	The size of the y component of the screen
<code>_subGoal- Number</code>	The initial number of sample points for the global planner
<code>_screen</code>	The PyGame screen that the <a href="#">PRMGenerator</a> will draw to

Definition at line 38 of file prm.py.

### 6.9.3 Member Function Documentation

#### 6.9.3.1 `def prm.PRMGenerator.draw ( self )`

Draws the graph.

Definition at line 306 of file prm.py.

#### 6.9.3.2 `def prm.PRMGenerator.drawPath ( self )`

Draws the selected shortest path.

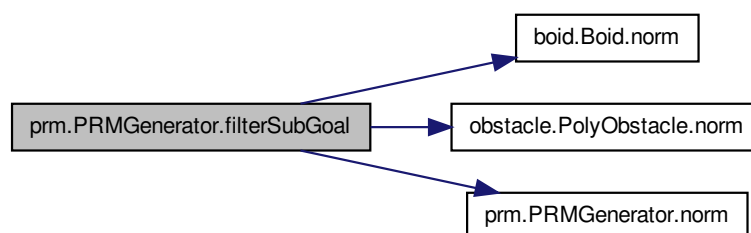
Definition at line 330 of file prm.py.

#### 6.9.3.3 `def prm.PRMGenerator.filterSubGoal ( self )`

Filters out sample points that are inside of obstacles or otherwise inadequate.

Definition at line 135 of file prm.py.

Here is the call graph for this function:

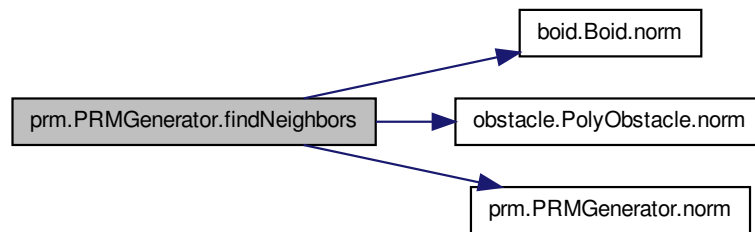


#### 6.9.3.4 `def prm.PRMGenerator.findNeighbors ( self, point )`

Finds suitable neighbours for a sample point.

Definition at line 158 of file prm.py.

Here is the call graph for this function:



#### 6.9.3.5 `def prm.PRMGenerator.generate ( self, subGoalRadius )`

Generates a series of random points that will become the roadmap and connects them and weights them into a graph.

If the goal and the starting point are not connected, more points are added. The roadmap is then searched for the shortest weighted distance which become the intermediate goals.

##### Parameters

<i>subGoalRadius</i>	The radius of the intermediate goals
----------------------	--------------------------------------

##### Returns

A list of sub goals from the roadmap connecting the starting point and the end goal

Definition at line 223 of file prm.py.

#### 6.9.3.6 `def prm.PRMGenerator.generatePositionList ( self, num )`

Generates the random positions for the sample points.

##### Parameters

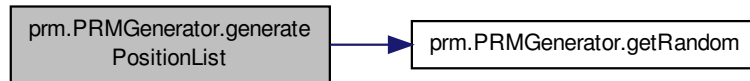
<i>num</i>	The number of points to generate
------------	----------------------------------

**Returns**

A list of random subgoals (sample points)

Definition at line 105 of file prm.py.

Here is the call graph for this function:

**6.9.3.7 def prm.PRMGenerator.getRandom ( self, p, q )**

Gets a random number and catches the ValueError if the two numbers are the same.

**Parameters**

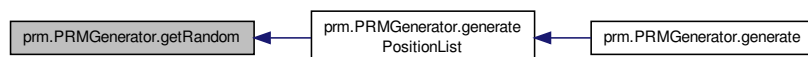
$p$	Lower bound for the random number
$q$	upper bound for the random number

**Returns**

A random number

Definition at line 206 of file prm.py.

Here is the caller graph for this function:

**6.9.3.8 def prm.PRMGenerator.getShortestPath ( self, roadmap, fromNode, toNode )**

Definition at line 295 of file prm.py.

Here is the call graph for this function:

**6.9.3.9 def prm.PRMGenerator.initOmega ( self, posList )**

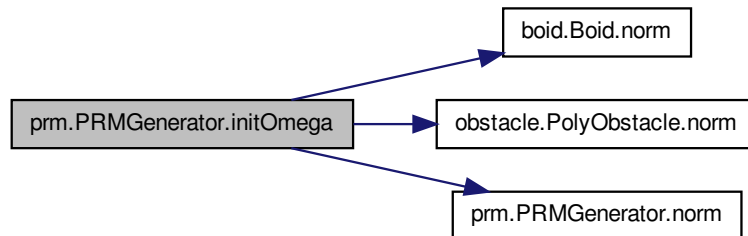
Initiates the omega function which holds the node weights.

## Parameters

<i>posList</i>	The list of positions for the sample points
----------------	---

Definition at line 118 of file prm.py.

Here is the call graph for this function:



#### 6.9.3.10 def prm.PRMGenerator.norm ( self, p1, p2 )

Gets the distance between p1 and p2.

## Parameters

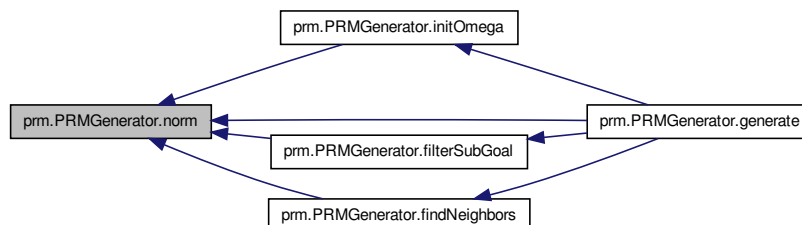
<i>p1</i>	The first point
<i>p2</i>	The second point

## Returns

The Eulidean distance from p1 to p2

Definition at line 96 of file prm.py.

Here is the caller graph for this function:



## 6.9.4 Member Data Documentation

### 6.9.4.1 prm.PRMGenerator.adjacentThresh

Distance that the PRM is willing to check when connecting sample points.

Definition at line 60 of file prm.py.

#### 6.9.4.2 prm.PRMGenerator.dontDraw

Definition at line 226 of file prm.py.

#### 6.9.4.3 prm.PRMGenerator.endPos

Position of the last goal.

Definition at line 47 of file prm.py.

#### 6.9.4.4 prm.PRMGenerator.goalNodes

Indexes of the goal positions.

Definition at line 82 of file prm.py.

#### 6.9.4.5 prm.PRMGenerator.gPosList

Holds the positions of the intermediate goals that were selected by the global path planner.

Definition at line 79 of file prm.py.

#### 6.9.4.6 prm.PRMGenerator.numNext

Maximum number of sample points that can be connected.

Definition at line 63 of file prm.py.

#### 6.9.4.7 prm.PRMGenerator.obstacleList

List of obstacles.

Definition at line 41 of file prm.py.

#### 6.9.4.8 prm.PRMGenerator.omegaDict

Dictionary (for easy access) that holds the weights for the nodes.

Definition at line 84 of file prm.py.

#### 6.9.4.9 prm.PRMGenerator.roadmap

The global roadmap.

It will be a graph represented as a dictionary

Definition at line 75 of file prm.py.

#### 6.9.4.10 prm.PRMGenerator.screen

PyGame screen that is will be drawn on.

Definition at line 50 of file prm.py.

**6.9.4.11** `prm.PRMGenerator.startPos`

Position of the first goal.

Definition at line 44 of file `prm.py`.

**6.9.4.12** `prm.PRMGenerator.subGoalNumber`

Number of initial sample points.

Definition at line 66 of file `prm.py`.

**6.9.4.13** `prm.PRMGenerator.subGoalPositionList`

Initial positions of the sample points.

Definition at line 69 of file `prm.py`.

**6.9.4.14** `prm.PRMGenerator.xSize`

Horizontal size of the PyGame screen.

Definition at line 53 of file `prm.py`.

**6.9.4.15** `prm.PRMGenerator.ySize`

Vertical size of the PyGame screen.

Definition at line 56 of file `prm.py`.

The documentation for this class was generated from the following file:

- `code/prm.py`



## Chapter 7

# File Documentation

### 7.1 code/boid.py File Reference

#### Classes

- class `boid.Boid`  
*Class which represents one boid.*

#### Namespaces

- `boid`

#### Constant Groups

- `boid`

#### Functions

- def `boid.guassianFunc`  
*Gamma function used to give a probability distribution of the flock in order to choose an appropriate neighbour.*

#### Variables

- string `boid.__author__` = "Alex Wallar <aw204@st-andrews.ac.uk>"

### 7.2 code/boidsimulation.py File Reference

#### Classes

- class `boidsimulation.FlockSim`  
*Main class for that is used for the simulation and display of the flock.*

#### Namespaces

- `boidsimulation`

## Constant Groups

- [boidsimulation](#)

## Variables

- string [boidsimulation.\\_\\_author\\_\\_](#) = "Alex Wallar <aw204@st-andrews.ac.uk>"

## 7.3 code/configuration.py File Reference

### Classes

- class [configuration.Configuration](#)  
*Static class that holds important global variables.*
- class [configuration.PolyFileConfiguration](#)  
*Extends the [Configuration](#) class.*

### Namespaces

- [configuration](#)

## Constant Groups

- [configuration](#)

## 7.4 code/dijkstra.py File Reference

### Namespaces

- [dijkstra](#)

## Constant Groups

- [dijkstra](#)

### Functions

- def [dijkstra.Dijkstra](#)  
*Find shortest paths from the start vertex to all vertices nearer than or equal to the end.*
- def [dijkstra.shortestPath](#)  
*Find a single shortest path from the given start vertex to the given end vertex.*

## 7.5 code/gatherstats.py File Reference

### Namespaces

- [gatherstats](#)

## Constant Groups

- [gatherstats](#)

## Functions

- def [gatherstats.generateStats](#)  
*Generates the statistics and saves them in a well known location.*

## Variables

- list [gatherstats.mapList](#)  
*A list of dictionaries used to store the map files, starting and ending points of the boids.*
- [gatherstats.testList](#) = mapList

## 7.6 code/goal.py File Reference

### Classes

- class [goal.CircleGoal](#)  
*Object that holds data about a goal modelled as circular configuration space.*

### Namespaces

- [goal](#)

### Constant Groups

- [goal](#)

### Variables

- string [goal.\\_\\_author\\_\\_](#) = "Alex Wallar <aw204@st-andrews.ac.uk>"

## 7.7 code/mapparser.py File Reference

### Namespaces

- [mapparser](#)

### Constant Groups

- [mapparser](#)

## Functions

- def [mapparser.mapVal](#)  
*Maps a value that is between `in_min` and `in_max` to a value between `out_min` and `out_max`.*
- def [mapparser.mparse](#)  
*Parses a map file into a list of obstacles.*

## 7.8 code/obstacle.py File Reference

### Classes

- class [obstacle.PolyObstacle](#)  
*Object that represents the an obstacle represented by a series of points (in the node list) which make up a set of lines.*

### Namespaces

- [obstacle](#)

### Constant Groups

- [obstacle](#)

### Variables

- string [obstacle.\\_\\_author\\_\\_](#) = "Alex Wallar <aw204@st-andrews.ac.uk>"

## 7.9 code/pridict.py File Reference

### Classes

- class [pridict.priorityDictionary](#)

### Namespaces

- [pridict](#)

### Constant Groups

- [pridict](#)

## 7.10 code/prm.py File Reference

### Classes

- class [prm.PRMGenerator](#)  
*Class used to hold methods and variables that are important for the global path planning problem.*

### Namespaces

- [prm](#)

### Constant Groups

- [prm](#)

## 7.11 code/test\_sim.py File Reference

### Namespaces

- [test\\_sim](#)

### Constant Groups

- [test\\_sim](#)

### Variables

- dictionary [test\\_sim.mapDict](#)  
*Main module to run for testing purposes.*
- tuple [test\\_sim.flockSize](#) = int(sys.argv[2])
- list [test\\_sim.startPoint](#) = mapDict[sys.argv[1]]
- list [test\\_sim.endPoint](#) = mapDict[sys.argv[1]]
- list [test\\_sim.mapFilePath](#) = sys.argv[1]
- [test\\_sim.obstacleFilePath](#) = None
- [test\\_sim.dynamicObstacleAutoGenerate](#) = False
- int [test\\_sim.generateTarget](#) = 0
- list [test\\_sim.arg](#) = sys.argv[3]
- tuple [test\\_sim.fs](#)

# Index

- `__author__`
    - `boid`, 10
    - `boidsimulation`, 10
    - `goal`, 14
    - `obstacle`, 15
  - `__init__`
    - `boid::Boid`, 22
    - `boidsimulation::FlockSim`, 44
    - `goal::CircleGoal`, 39
    - `obstacle::PolyObstacle`, 54
    - `pridict::priorityDictionary`, 62
    - `prm::PRMGenerator`, 64
  - `__iter__`
    - `pridict::priorityDictionary`, 62
  - `__setitem__`
    - `pridict::priorityDictionary`, 62
- `adjacentThresh`
  - `prm::PRMGenerator`, 68
- `animate`
  - `boidsimulation::FlockSim`, 44
- `arg`
  - `test_sim`, 16
- `auto_gen_number`
  - `boidsimulation::FlockSim`, 47
  - `configuration::PolyFileConfiguration`, 51
- `auto_gen_obst`
  - `boidsimulation::FlockSim`, 47
  - `configuration::PolyFileConfiguration`, 51
- `autoGenerateDynamicObstacles`
  - `configuration::PolyFileConfiguration`, 51
- `avg`
  - `boidsimulation::FlockSim`, 44
- `avgPoint`
  - `obstacle::PolyObstacle`, 60
- `bAlpha`
  - `boid::Boid`, 33
- `bBeta`
  - `boid::Boid`, 33
- `bConst`
  - `boid::Boid`, 33
- `bDelta`
  - `boid::Boid`, 33
- `bInfluenceR`
  - `boid::Boid`, 34
- `BLACK`
  - `boidsimulation::FlockSim`, 47
- `boid`, 9
  - `__author__`, 10
  - `gaussianFunc`, 9
- `boid.Boid`, 19
- `boid::Boid`
  - `__init__`, 22
  - `bAlpha`, 33
  - `bBeta`, 33
  - `bConst`, 33
  - `bDelta`, 33
  - `bInfluenceR`, 34
  - `boidList`, 34
  - `color`, 34
  - `compWeightList`, 34
  - `determineNewPath`, 22
  - `determineRandomWalk`, 22
  - `dim`, 34
  - `draw`, 22
  - `ePos`, 34
  - `endIndex`, 34
  - `findMax`, 23
  - `gAlpha`, 34
  - `gBeta`, 34
  - `gConst`, 35
  - `gDelta`, 35
  - `gammaFunc`, 34
  - `getBoidVectorList`, 23
  - `getDirectionVector`, 24
  - `getGoalVector`, 24
  - `getNeighborVectorList`, 25
  - `getObstacleVectorList`, 26
  - `getVar`, 27
  - `goal`, 35
  - `goalCounter`, 35
  - `goalList`, 35
  - `goalNodes`, 35
  - `headWeightList`, 35
  - `heading`, 35
  - `inGoal`, 27
  - `inWorld`, 28
  - `initFunctionParameters`, 28
  - `mag`, 28
  - `nStuckDAvg`, 36
  - `nStuckDSigma`, 36
  - `neighborSize`, 35
  - `norm`, 29
  - `obBeta`, 36
  - `obInfluenceR`, 36
  - `obstacleFunc`, 29
  - `obstacleList`, 36
  - `pointAllowed`, 30

- position, 36
- positionBuffer, 36
- prmGen, 36
- radius, 36
- randWalkCount, 37
- randomWalkX, 37
- randomWalkY, 37
- reduceWeightValues, 30
- roadmap, 37
- sPos, 37
- screen, 37
- setBoidList, 31
- setNewGoal, 31
- sigmoidFunc, 31
- speed, 37
- stuck, 37
- stuckConst, 37
- stuckDAvg, 38
- stuckDSigma, 38
- sumDivide, 32
- update, 32
- updatePositionBuffer, 33
- ySize, 38
- boidList
  - boid::Boid, 34
  - configuration::PolyFileConfiguration, 51
- boidSpeed
  - configuration::Configuration, 41
- boidsimulation, 10
  - \_\_author\_\_, 10
- boidsimulation.FlockSim, 42
- boidsimulation::FlockSim
  - \_\_init\_\_, 44
  - animate, 44
  - auto\_gen\_number, 47
  - auto\_gen\_obst, 47
  - avg, 44
  - BLACK, 47
  - config, 47
  - counter, 47
  - dataFile, 47
  - dim, 47
  - done, 47
  - ePos, 48
  - flockSize, 48
  - font, 48
  - frameCounter, 48
  - getBoidData, 45
  - getStats, 45
  - init\_prm, 45
  - iterations, 48
  - mapFile, 48
  - numInGoal, 48
  - obstacleFile, 48
  - play, 46
  - render, 46
  - sPos, 48
  - startTime, 49
  - surfaceList, 49
  - WHITE, 49
- boundary
  - obstacle::PolyObstacle, 60
- change\_direction
  - obstacle::PolyObstacle, 54
- checkCollisionWithOtherObstacles
  - obstacle::PolyObstacle, 54
- code/boid.py, 71
- code/boidsimulation.py, 71
- code/configuration.py, 72
- code/dijkstra.py, 72
- code/gatherstats.py, 72
- code/goal.py, 73
- code/mapparser.py, 73
- code/obstacle.py, 74
- code/priodict.py, 74
- code/prm.py, 74
- code/test\_sim.py, 75
- color
  - boid::Boid, 34
- colorList
  - configuration::Configuration, 41
- colors
  - goal::CircleGoal, 39
  - obstacle::PolyObstacle, 60
- compWeightList
  - boid::Boid, 34
- config
  - boidsimulation::FlockSim, 47
- configuration, 10
- configuration.Configuration, 40
- configuration.PolyFileConfiguration, 49
- configuration::Configuration
  - boidSpeed, 41
  - colorList, 41
  - dim, 41
  - goalRadius, 41
  - numNeighbours, 41
  - numSamplePoints, 41
  - screen, 41
- configuration::PolyFileConfiguration
  - auto\_gen\_number, 51
  - auto\_gen\_obst, 51
  - autoGenerateDynamicObstacles, 51
  - boidList, 51
  - endPoint, 51
  - goalList, 52
  - initVars, 51
  - nodes, 52
  - obstacleList, 52
  - parseDynamicObstacles, 51
  - prmGen, 52
  - startPoint, 52
- counter
  - boidsimulation::FlockSim, 47
- dataFile

- boidsimulation::FlockSim, 47
- detectCollision
  - obstacle::PolyObstacle, 54
- determine\_last\_direction
  - obstacle::PolyObstacle, 54
- determineNewPath
  - boid::Boid, 22
- determineRandomWalk
  - boid::Boid, 22
- dict, 42
- Dijkstra
  - dijkstra, 11
- dijkstra, 10
  - Dijkstra, 11
  - shortestPath, 12
- dim
  - boid::Boid, 34
  - boidsimulation::FlockSim, 47
  - configuration::Configuration, 41
- displacement
  - obstacle::PolyObstacle, 60
- done
  - boidsimulation::FlockSim, 47
- dontDraw
  - prm::PRMGenerator, 69
- draw
  - boid::Boid, 22
  - goal::CircleGoal, 39
  - obstacle::PolyObstacle, 55
  - prm::PRMGenerator, 65
- drawPath
  - prm::PRMGenerator, 65
- dynamic
  - obstacle::PolyObstacle, 60
- dynamicObstacleAutoGenerate
  - test\_sim, 16
- ePos
  - boid::Boid, 34
  - boidsimulation::FlockSim, 48
- endIndex
  - boid::Boid, 34
- endPoint
  - configuration::PolyFileConfiguration, 51
  - test\_sim, 16
- endPos
  - prm::PRMGenerator, 69
- estimatePoly
  - obstacle::PolyObstacle, 55
- filterSubGoal
  - prm::PRMGenerator, 65
- findMax
  - boid::Boid, 23
- findNeighbors
  - prm::PRMGenerator, 65
- flockSize
  - boidsimulation::FlockSim, 48
  - test\_sim, 16
- font
  - boidsimulation::FlockSim, 48
- frameCounter
  - boidsimulation::FlockSim, 48
- fs
  - test\_sim, 16
- gAlpha
  - boid::Boid, 34
- gBeta
  - boid::Boid, 34
- gConst
  - boid::Boid, 35
- gDelta
  - boid::Boid, 35
- gPosList
  - prm::PRMGenerator, 69
- gammaFunc
  - boid::Boid, 34
- gatherstats, 13
  - generateStats, 13
  - mapList, 13
  - testList, 13
- generate
  - prm::PRMGenerator, 66
- generatePositionList
  - prm::PRMGenerator, 66
- generateStats
  - gatherstats, 13
- generateTarget
  - test\_sim, 16
- getBoidData
  - boidsimulation::FlockSim, 45
- getBoidVectorList
  - boid::Boid, 23
- getClosestPoint
  - obstacle::PolyObstacle, 55
- getDirectionVector
  - boid::Boid, 24
- getGoalVector
  - boid::Boid, 24
- getNeighborVectorList
  - boid::Boid, 25
- getObstacleVectorList
  - boid::Boid, 26
- getPoint
  - obstacle::PolyObstacle, 56
- getRadius
  - obstacle::PolyObstacle, 56
- getRandom
  - prm::PRMGenerator, 67
- getShortestPath
  - prm::PRMGenerator, 67
- getStats
  - boidsimulation::FlockSim, 45
- getVar
  - boid::Boid, 27
- goal, 13
  - \_\_author\_\_, 14



- boid::Boid, 35
- goal.CircleGoal, 38
- goal::CircleGoal
  - \_\_init\_\_, 39
  - colors, 39
  - draw, 39
  - position, 39
  - radius, 39
  - screen, 39
- goalCounter
  - boid::Boid, 35
- goalList
  - boid::Boid, 35
  - configuration::PolyFileConfiguration, 52
- goalNodes
  - boid::Boid, 35
  - prm::PRMGenerator, 69
- goalRadius
  - configuration::Configuration, 41
- gaussianFunc
  - boid, 9
- headWeightList
  - boid::Boid, 35
- heading
  - boid::Boid, 35
- inGoal
  - boid::Boid, 27
- inWorld
  - boid::Boid, 28
- init\_prm
  - boidsimulation::FlockSim, 45
- initFunctionParameters
  - boid::Boid, 28
- initOmega
  - prm::PRMGenerator, 67
- initVars
  - configuration::PolyFileConfiguration, 51
- iterations
  - boidsimulation::FlockSim, 48
- mag
  - boid::Boid, 28
- mapDict
  - test\_sim, 16
- mapFile
  - boidsimulation::FlockSim, 48
- mapFilePath
  - test\_sim, 17
- mapList
  - gatherstats, 13
- mapVal
  - mapparser, 14
- mapparser, 14
  - mapVal, 14
  - mparse, 14
- max\_displacement
  - obstacle::PolyObstacle, 60
- maxDist
  - obstacle::PolyObstacle, 60
- mparse
  - mapparser, 14
- nStuckDAvg
  - boid::Boid, 36
- nStuckDSigma
  - boid::Boid, 36
- neighborSize
  - boid::Boid, 35
- nodes
  - configuration::PolyFileConfiguration, 52
  - obstacle::PolyObstacle, 60
- norm
  - boid::Boid, 29
  - obstacle::PolyObstacle, 57
  - prm::PRMGenerator, 68
- numInGoal
  - boidsimulation::FlockSim, 48
- numNeighbours
  - configuration::Configuration, 41
- numNext
  - prm::PRMGenerator, 69
- numSamplePoints
  - configuration::Configuration, 41
- obBeta
  - boid::Boid, 36
- obInfluenceR
  - boid::Boid, 36
- obstacle, 15
  - \_\_author\_\_, 15
- obstacle.PolyObstacle, 52
- obstacle::PolyObstacle
  - \_\_init\_\_, 54
  - avgPoint, 60
  - boundary, 60
  - change\_direction, 54
  - checkCollisionWithOtherObstacles, 54
  - colors, 60
  - detectCollision, 54
  - determine\_last\_direction, 54
  - displacement, 60
  - draw, 55
  - dynamic, 60
  - estimatePoly, 55
  - getClosestPoint, 55
  - getPoint, 56
  - getRadius, 56
  - max\_displacement, 60
  - maxDist, 60
  - nodes, 60
  - norm, 57
  - obstacles, 60
  - pointAllowed, 57
  - pointInPoly, 58
  - rayintersectseg, 59
  - removeSelfFromObstacleList, 59

- screen, 60
- translate, 59
- velocity, 61
- obstacleFile
  - boidsimulation::FlockSim, 48
- obstacleFilePath
  - test\_sim, 17
- obstacleFunc
  - boid::Boid, 29
- obstacleList
  - boid::Boid, 36
  - configuration::PolyFileConfiguration, 52
  - prm::PRMGenerator, 69
- obstacles
  - obstacle::PolyObstacle, 60
- omegaDict
  - prm::PRMGenerator, 69
- parseDynamicObstacles
  - configuration::PolyFileConfiguration, 51
- play
  - boidsimulation::FlockSim, 46
- pointAllowed
  - boid::Boid, 30
  - obstacle::PolyObstacle, 57
- pointInPoly
  - obstacle::PolyObstacle, 58
- position
  - boid::Boid, 36
  - goal::CircleGoal, 39
- positionBuffer
  - boid::Boid, 36
- pridict, 15
- pridict.priorityDictionary, 61
- pridict::priorityDictionary
  - \_\_init\_\_, 62
  - \_\_iter\_\_, 62
  - \_\_setitem\_\_, 62
  - setdefault, 62
  - smallest, 63
- prm, 15
- prm.PRMGenerator, 63
- prm::PRMGenerator
  - \_\_init\_\_, 64
  - adjacentThresh, 68
  - dontDraw, 69
  - draw, 65
  - drawPath, 65
  - endPos, 69
  - filterSubGoal, 65
  - findNeighbors, 65
  - gPosList, 69
  - generate, 66
  - generatePositionList, 66
  - getRandom, 67
  - getShortestPath, 67
  - goalNodes, 69
  - initOmega, 67
  - norm, 68
  - numNext, 69
  - obstacleList, 69
  - omegaDict, 69
  - roadmap, 69
  - screen, 69
  - startPos, 69
  - subGoalNumber, 70
  - subGoalPositionList, 70
  - xSize, 70
  - ySize, 70
- prmGen
  - boid::Boid, 36
  - configuration::PolyFileConfiguration, 52
- radius
  - boid::Boid, 36
  - goal::CircleGoal, 39
- randWalkCount
  - boid::Boid, 37
- randomWalkX
  - boid::Boid, 37
- randomWalkY
  - boid::Boid, 37
- rayintersectseg
  - obstacle::PolyObstacle, 59
- reduceWeightValues
  - boid::Boid, 30
- removeSelfFromObstacleList
  - obstacle::PolyObstacle, 59
- render
  - boidsimulation::FlockSim, 46
- roadmap
  - boid::Boid, 37
  - prm::PRMGenerator, 69
- sPos
  - boid::Boid, 37
  - boidsimulation::FlockSim, 48
- screen
  - boid::Boid, 37
  - configuration::Configuration, 41
  - goal::CircleGoal, 39
  - obstacle::PolyObstacle, 60
  - prm::PRMGenerator, 69
- setBoidList
  - boid::Boid, 31
- setNewGoal
  - boid::Boid, 31
- setdefault
  - pridict::priorityDictionary, 62
- shortestPath
  - dijkstra, 12
- sigmoidFunc
  - boid::Boid, 31
- smallest
  - pridict::priorityDictionary, 63
- speed
  - boid::Boid, 37
- startPoint

- configuration::PolyFileConfiguration, [52](#)
  - test\_sim, [17](#)
- startPos
  - prm::PRMGenerator, [69](#)
- startTime
  - boidsimulation::FlockSim, [49](#)
- stuck
  - boid::Boid, [37](#)
- stuckConst
  - boid::Boid, [37](#)
- stuckDAvg
  - boid::Boid, [38](#)
- stuckDSigma
  - boid::Boid, [38](#)
- subGoalNumber
  - prm::PRMGenerator, [70](#)
- subGoalPositionList
  - prm::PRMGenerator, [70](#)
- sumDivide
  - boid::Boid, [32](#)
- surfaceList
  - boidsimulation::FlockSim, [49](#)
- test\_sim, [16](#)
  - arg, [16](#)
  - dynamicObstacleAutoGenerate, [16](#)
  - endPoint, [16](#)
  - flockSize, [16](#)
  - fs, [16](#)
  - generateTarget, [16](#)
  - mapDict, [16](#)
  - mapFilePath, [17](#)
  - obstacleFilePath, [17](#)
  - startPoint, [17](#)
- testList
  - gatherstats, [13](#)
- translate
  - obstacle::PolyObstacle, [59](#)
- update
  - boid::Boid, [32](#)
- updatePositionBuffer
  - boid::Boid, [33](#)
- velocity
  - obstacle::PolyObstacle, [61](#)
- WHITE
  - boidsimulation::FlockSim, [49](#)
- xSize
  - prm::PRMGenerator, [70](#)
- ySize
  - boid::Boid, [38](#)
  - prm::PRMGenerator, [70](#)