worldGraph class

worldGraph represents the world as a graph: crossings are vertices and lines are edges. By specifying the starting position and orientation, as well as the target position, the world graph is able to provide navigation by calling the getDirection() method. After each call to getDirection(), the caller is expected to specify whether the last instruction succeeded or not using the instructionCompleted() or instructionFailed() methods.

private variables

_graph

graph holds the data structure that represents the whole graph (vertices and edges)

_target

target holds the target initially passed in the constructor method

_position

position holds the current position the robot is in, and is initiated in the constructor with the starting position

_orientation

orientation holds the current orientation of the robot, and is initiated in the constructor with the starting orientation

_instructedDirection

instructed direction holds the last direction returned by the getDirection method

methods

worldGraph(int graphSize, ii position, ii target, orientationTypedef initialOrientation)

position and target are the starting and target positions; initialOrientation is the starting orientation of the robot; graphSize is the size of the graph. Bellow an example of size 4 world graph:

```
O O O O
  | |
O-O-O-O
  | |
O-O-O-O
  | |
O O O O
```

direction getDirection()

getDirection returns an instruction in the form of a direction to turn and then follow the line until reaching the next vertex or finding an obstacle in the middle.

void instructionCompleted()

instructionCompleted informs the world graph that the last instruction given by getDirection could be completed successfully (i.e. following the line specified by getDirection led to a vertex). It's assumed that once the robot reached its destination it maintained the orientation of the guide line just traveled.

void instructionFailed()

instructionFailed informs the world graph that the last instruction given by getDirection couldn't be completed (i.e there was an obstacle in the way). It's assumed that once encountering an obstacle, the robot turns 180 degrees and returns to the last visited vertex, maintaining its orientation upon arrival.