public class Wizard implements RobotDriver

The Wizard Class implements the RobotDriver interface, the Wizard object is created inside the MazeController file before the BasicRobot is passed to it. The wizard algorithm is designed to keep the robot moving into the next position that is closer to the exit when compared to its current position. The wizard algorithm follows the show solution line to the exit. It is known as the most effective algorithm due to its ability to cheat.

checkForExit() is designed to check all 4 directions and see if the robot is next to the maze exit if it is then it calls setEndGame().

move() is designed to move the robot forward one cell. It then calls notifyViewerRedraw() and puts the thread to sleep.

turnLeftMove() is designed to turn the robot left and move the robot forward one cell. It then calls notifyViewerRedraw() and puts the thread to sleep.

turnRightMove() is designed to turn the robot right and move the robot forward one cell. It then calls notifyViewerRedraw() and puts the thread to sleep.

turnAround() is designed to turn the robot. It then calls notifyViewerRedraw() and puts the thread to sleep.

wizardRight() is designed to get the current position of the robot and the direction its facing and with that information it determines if turning right and moving will bring it closer to the exit. It makes this calculation by checking if the future location will have a lower distance number when compared to the current location if it will it goes ahead and calls the turnRightMove().

wizardLeft() is designed to get the current position of the robot and the direction its facing and with that information it determines if turning left and moving will bring it closer to the exit. It makes this calculation by checking if the future location will have a lower distance number when compared to the current location if it will it goes ahead and calls the turnLeftMove().

setEndGame() is designed to change the maze's state to end game then to call the notifyViewerRedraw().

Collaborators: RobotDriver, BasicRobot, and MazeController

@authors Chris Wolinski & Marcelino Dayrit

---

public class Pledge implements RobotDriver

The Pledge Class implements the RobotDriver interface, the Pledge object is created inside the MazeController file before the BasicRobot is passed to it. The pledge algorithm is designed to keep the robot moving forward if the counter is equal to zero and glued to the wall on its right side if the counter is not equal to zero, the counter decrements by 1 if the robot takes a left turn and increments by 1 if the robot takes a right turn. The pledge algorithm bypasses the problem with the wallFollower algorithm because it is able to leave the possible obstacle created by rooms thanks to the counter.

checkForExit() is designed to check all 4 directions and see if the robot is next to the maze exit if it is then it calls setEndGame().

move() is designed to move the robot forward one cell.  It then calls notifyViewerRedraw() and puts the thread to sleep.

turnLeft() is designed to turn the robot right and decrement the counter by 1.  It then calls notifyViewerRedraw() and puts the thread to sleep.

turnRight() is designed to turn the robot right and increment the counter by 1. It then calls notifyViewerRedraw() and puts the thread to sleep.

setEndGame() is designed to change the maze's state to end game then to call the notifyViewerRedraw().

Collaborators: RobotDriver, BasicRobot, and MazeController

@authors Chris Wolinski & Marcelino Dayrit

---

public class WallFollower implements RobotDriver

The WallFollower Class implements the RobotDriver interface, the WallFollower object is created inside the MazeController file before the BasicRobot is passed to it. The wallFollower algorithm is designed to keep the robot glued to the wall on its left side and follow it to the exit location. The only possible problem with this algorithm is it starts in a room that causes the function follow a wall that is an island and is disconnected from the rest of the maze causing the algorithm to loop until it runs out of battery.

checkForExit() is designed to check all 4 directions and see if the robot is next to the maze exit if it is then it calls setEndGame().

move() is designed to move the robot forward one cell. It then calls notifyViewerRedraw() and puts the thread to sleep.

turnLeftMove() is designed to turn the robot left and move the robot forward one cell. It then calls notifyViewerRedraw() and puts the thread to sleep.

turnRightMove() is designed to turn the robot right and move the robot forward one cell. It then calls notifyViewerRedraw() and puts the thread to sleep.

turnAroundMove() is designed to turn the robot around and move the robot forward one cell. It then calls notifyViewerRedraw() and puts the thread to sleep.

setEndGame() is designed to change the maze's state to end game then to call the notifyViewerRedraw().

Collaborators: RobotDriver, BasicRobot, and MazeController

@authors Chris Wolinski & Marcelino Dayrit