Laser Maze Assignment Summary

Jingran Wen

Jingran.wen@utah.edu

This document is used to demonstrate a Java solution for a simple Laser Maze game.

Table of Contents

[The logic to solve a Laser Maze 2](#_Toc498271074)

[Requirements and Assumptions 2](#_Toc498271075)

[Project Design 2](#_Toc498271076)

[Unit Test 2](#_Toc498271077)

[Lasermaze Java Package Overview 3](#_Toc498271078)

[Runnable jar file: maze.jar 4](#_Toc498271079)

[Git 4](#_Toc498271080)

[Project Documentation 4](#_Toc498271081)

[What is in the zip file I send? 4](#_Toc498271082)

[File logs 4](#_Toc498271083)

# The logic to solve a Laser Maze

Laser light travels in the maze grid. If light hit a mirror, its travel direction will change, otherwise it travels the same direction. Once the light is outside the maze grid, the maze game is solved. The light might travel in loops in the maze, in which case the light is trapped inside the maze. If the light enter one square of the maze with the same direction twice, we can tell that there is a loop in the travel path.

# Requirements and Assumptions

The project follows the requirements and assumptions given in the instruction document - DataScientistHW-LaserMaze3\_updated1.pdf.

To simplify the question, there are some assumptions: 1) **both side** of all the mirrors can change the direction of laser light; 2) mirrors can only change the direction of laser light by 90 degree; 3) only two type of mirrors (\ and /); 4) starting point can only be in the empty grid.

More requirements and assumptions are listed in the instruction document.

# Project Design

This project follows the object-oriented design. Interface is used to make sure that this project is extendable and friendly to project developers. Two Interfaces are designed in this project, and they are Maze and Mirror. The Maze interface defines functions used to solve a Laser Maze, e.g. how to define there is a loop in the maze. Mirror defines functions which represents the behavior of a mirror, e.g. how mirror change the travel directions of a light. Developers can implement the game solutions in their own way using these interfaces.

Class MyMirror is my implantation of the Mirror interface. This class defines the variables and behaviors of a mirror. Variables and functions related to a mirror can be added in this class, e.g. new rules added to the mirror, or new types of the mirror is added.

Class Status is used to record the status of laser light. Right now, it has variables to record the coordination of a square the light hit and the direction of the light when it enters this square.

Class MazeParameters is used to record the parameters used to initialize a game, including the size of the maze, the starting points, and location and type of mirrors.

Class MyMaze is my implementation of the Maze interface. MyMaze records variables that are used to initialize and solve the puzzle, as well as the request results. Examples of the variables are initialization parameters of a maze puzzle, the number of squares traveled, coordination of the final square if there is one, and the path of the light in the maze. It implements functions on how to extract maze parameters from input file, how to solve a Laser Maze, how the light travels in the maze, and how to write the output into a file.

# Unit Test

JUint 4 is used to do unit test. Almost all the functions in each class are tested with different test cases to sure that they covered the situation of normal and corner cases. A test suite is also created to bundle all the unit tests.

# Lasermaze Java Package Overview

The lasermaze package includes 2 Interfaces and 4 main Classes, together to represent the objects, variables, and methods used to solve the Laser Maze game. The main() function is in Main class. Interfaces and main classes are under src folder. The package also includes 8 Junit tests, which names are started with Test\*, and a test suite class (AllTests). The test codes are under test folder. Table 1 and Table 2 are a brief summary of each Interface and Class. The detailed description of each classes can be found in the documentation of the project under the doc folder.

Table 1 Interface Summary

|  |  |
| --- | --- |
| **Interface** | **Description** |
| [**Maze**](file:///C:\Users\Jingran\git\myLocalLaserMaze\LaserMaze\doc\lasermaze\Maze.html) | The Maze interface defines functions used to solve a Laser Maze game. |
| [**Mirror**](file:///C:\Users\Jingran\git\myLocalLaserMaze\LaserMaze\doc\lasermaze\Mirror.html) | This Mirror interface defines functions on the behavior and characteristics of a mirror. |

Table 2 Classes Summary

|  |  |
| --- | --- |
| **Class** | **Description** |
| [**AllTests**](file:///C:\Users\Jingran\git\myLocalLaserMaze\LaserMaze\doc\lasermaze\AllTests.html) | Test suite. |
| [**Main**](file:///C:\Users\Jingran\git\myLocalLaserMaze\LaserMaze\doc\lasermaze\Main.html) | The main() function is included in this class. |
| [**MazeParameters**](file:///C:\Users\Jingran\git\myLocalLaserMaze\LaserMaze\doc\lasermaze\MazeParameters.html) | This MazeParameters class is used to represent the initial parameters of a Laser Maze game. |
| [**MyMaze**](file:///C:\Users\Jingran\git\myLocalLaserMaze\LaserMaze\doc\lasermaze\MyMaze.html) | This MyMaze class is an implementation of Maze interface. |
| [**MyMirror**](file:///C:\Users\Jingran\git\myLocalLaserMaze\LaserMaze\doc\lasermaze\MyMirror.html) | This MyMirror class is an implementation of Mirror interface. |
| [**Status**](file:///C:\Users\Jingran\git\myLocalLaserMaze\LaserMaze\doc\lasermaze\Status.html) | This class is used to record the travel status of laser light. |
| [**TestGetNextStatus**](file:///C:\Users\Jingran\git\myLocalLaserMaze\LaserMaze\doc\lasermaze\TestGetNextStatus.html) | Test getNextStep function of MyMaze class. |
| [**TestGetParameters**](file:///C:\Users\Jingran\git\myLocalLaserMaze\LaserMaze\doc\lasermaze\TestGetParameters.html) | Test getParameter function of MazeParameter class. |
| [**TestMazeParameterConstructor**](file:///C:\Users\Jingran\git\myLocalLaserMaze\LaserMaze\doc\lasermaze\TestMazeParameterConstructor.html) | Test constructor of MazeParameter Class. |
| [**TestMazeReadParametersFromFiles**](file:///C:\Users\Jingran\git\myLocalLaserMaze\LaserMaze\doc\lasermaze\TestMazeReadParametersFromFiles.html) | Test readParametersFromFile function of MyMazz class. |
| [**TestMirrorChangeDirections**](file:///C:\Users\Jingran\git\myLocalLaserMaze\LaserMaze\doc\lasermaze\TestMirrorChangeDirections.html) | Test changeDirections function of Mirror class. |
| [**TestOutOfBoundary**](file:///C:\Users\Jingran\git\myLocalLaserMaze\LaserMaze\doc\lasermaze\TestOutOfBoundary.html) | Test outOfBoundary function of MyMaze class. |
| [**TestSolveMaze**](file:///C:\Users\Jingran\git\myLocalLaserMaze\LaserMaze\doc\lasermaze\TestSolveMaze.html) | Test the solveMaze function of MyMaze class. |
| [**TestStatustEquals**](file:///C:\Users\Jingran\git\myLocalLaserMaze\LaserMaze\doc\lasermaze\TestStatustEquals.html) | Test equlas function of Status class. |

# Runnable jar file: maze.jar

maze.jar is a runnable program of the Laze Maze project. To run the program you need Java Runtime Environment installed. Please check this [link](http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html) to download JRE and install if it is not already installed in your computer.

To run maze.jar in the command line, switch to the path where maze.jar is in, and type:

maze.jar path\_to\_inputfile/input\_file\_name path\_to\_output\_file/output\_file\_name

If there are white spaces in the path and the filename, use double quote to quote the “file\_path/file\_name”. *Please use forward slash “/” in your path.*

maze.jar will solve the Laser Maze game using information given in input file and functions in the lasermaze package, and write the output into the output file. The requirement of input and output file are listed in the instruction document - DataScientistHW-LaserMaze3\_updated1.pdf. *If the input file is not in the required format, no output file will be generated.*

# Git

Git is used to track the change history of thi project. Git histories can be found in the .git folder. The lasermase Java package is also pushed to GitHub. You can get the latest version of this package at: https://github.com/leanawen/LaserMaze.

# Project Documentation

The detailed description of this project, Interface, Class, Variables, and Methods can be found in Javadoc documents in the doc folder. The documents are in html format, and you might need a web browser to read them properly.

# What is in the zip file I send?

* Source code: under myLocalLaserMaze/LaserMaze/src/lasermaze
* Test code: under myLocalLaserMaze/LaserMaze/test/lasermaze
* Runnable file: myLocalLaserMaze/maze.jar
* Git history: under myLocalLaserMaze/.git
* Project documentation: myLocalLaserMaze/LaserMaze/doc
* Test input files and expected output files: myLocalLaserMaze/LaserMaze/testcases
* The project summary document (this file): under myLocalLaserMaze/Laser Maze Assignment Summary.docx

# File logs

|  |  |  |
| --- | --- | --- |
| Date and Time | Authors | Notes |
| 11/12/2017 | Jingran Wen | First created this document. |
|  |  |  |