The Journal of Open Source Software

MDL Suite: A language, generator and compiler for describing mazes

Akash Nag[1]

1 Lecturer, Dept. of Computer Science, M.U.C. Women's College

DOI: 10.21105/joss.01815

Software

- Review
- Repository
- Archive

Editor: Vincent Knight

Reviewers:

- @rreinecke
- @gradvohl

Submitted: 11 October 2019
Published: 06 February 2020

Background

Mazes have fascinated people since ancient times, and extensive research has been undertaken to develop cutting-edge maze-solving algorithms that are faster or use lesser resources. The popular micromouse competitions since the late 1970s ("History - micromouse online," 2010) have provided much of the push behind such research. With each year, the competition is fiercer as new cutting-edge maze solving algorithms are invented.

Statement of Need

A typical problem in describing a new maze-solving algorithm in a journal article is to actually generate the maze image. Although random maze generators are widely available, often a paper needs to describe a new algorithm that optimizes certain types of paths and loops in a maze. In such cases, we need to have precise control over how the maze looks. Moreover, the paths taken by a robot (or solver) also needs to be marked out (e.g. see (Gupta & Sehgal, 2014)). To generate such images, the researcher needs to either painstakingly draw it or write a program to do so, both of which are time-consuming and tedious. Even if such an image could be generated for journal articles, they can not be shared among researchers so that they may try their own solvers on them, because these images cannot be readily converted into program-readable mazes. To solve the aforesaid problems, we have developed a markup language called Maze Description Language (MDL), which is human-readable yet terse.

State of the field

Existing maze generators can be categorized into four classes:

1. Those which generate mazes as text (e.g. as 0s and 1s, see ("GitHub - oppenheimj / maze-generator," 2018)).
2. Those which generate random mazes as images/PDF but have no solution paths marked out ("Maze generator," 2019).
3. Those which generate maze images/document-format and have solution paths marked out ("Maze generator and solver - wolfram demonstrations project," 2011).
4. Those which generate maze images/3D-models and can have partial paths marked out (Ayaz, Allen, Platek, & Onaral, 2008).

The generators in the first category are only suitable as inputs to some maze-solving algorithms and can not be displayed/published in a paper. The second category generators generate only the mazes and no paths can be marked out, making them unsuitable for researchers trying to describe the maze-solving process undertaken by their own proposed algorithms in a step-by-step fashion. The third category generators can only mark out solution paths, but researchers often need to display how some algorithm can take detours, be stuck in dead-ends, etc. The final class of generators is where our proposed tool falls in is suitable for research purposes. However, the ones that do exist currently (such as (Ayaz et al., 2008)) are not open-source. Moreover, the paths they depict in MazeViewer (part of MazeSuite) come from log-files generated by researchers. Therefore, the path-traversal data comes purely from physical experiments (with fine resolution) and not suitable for simply depicting path-traversal by an algorithm. Although such path-data (as expected by MazeViewer) can be generated programmtically, it would require more work on the part of the researcher.

Summary

With MDL-Suite, mazes can be easily described with minimal code, and visited paths can be marked out. Multiple visited paths can also be marked using separate colors. The mazes are fully customizable and a path-based description allows manual maze design very easy. The MDL suite contains two programs: a MDL compiler and a MDL generator.

The **MDL generator** is a random maze generator that, given the maze specifications, generates a random maze in MDL format.

The **MDL compiler** compiles code written in MDL, either manually written or generated using the MDL generator, into one of many output formats. The mazes can be compiled into images (jpg, png, tiff, bmp) or they can directly be converted into code for 2D arrays (int, boolean, char) for quick prototyping in Python or Java.

The MDL suite is a command-line utility written in Java and hence, requires a JVM to be installed in order to use it.

Key Features

1. The maze design is fully customizable with a number of options for setting colors, grids, coordinates, and cell-sizes.
2. Paths can be described minimally with a number of shorthands available, useful for manual maze design. Alternately, mazes can also be described row-wise, useful for algorithmically generated mazes.
3. Both generation and compilation commands require no complicated options or arguments except the input files and are therefore, very easy to use. Furthermore, both tools support multiple maze generation/compilation with a single command.
4. The MDL-format files are small text files and hence can be easily shared among researchers for collaboration. They can also be edited and/or converted to Java/Python code, or images using only a text-editor.
5. Research articles often require resizing images for better fit/placement and adjusting maze sizes (and hence image resolution) is quick and easy.
6. Multiple paths can be marked out in separate colors, and even sprites (images) can be used for marking those paths or the current position of the solver.

Examples

Below we see a manually generated maze with only 12 lines of MDL code (and compiled with the MDL compiler into png):

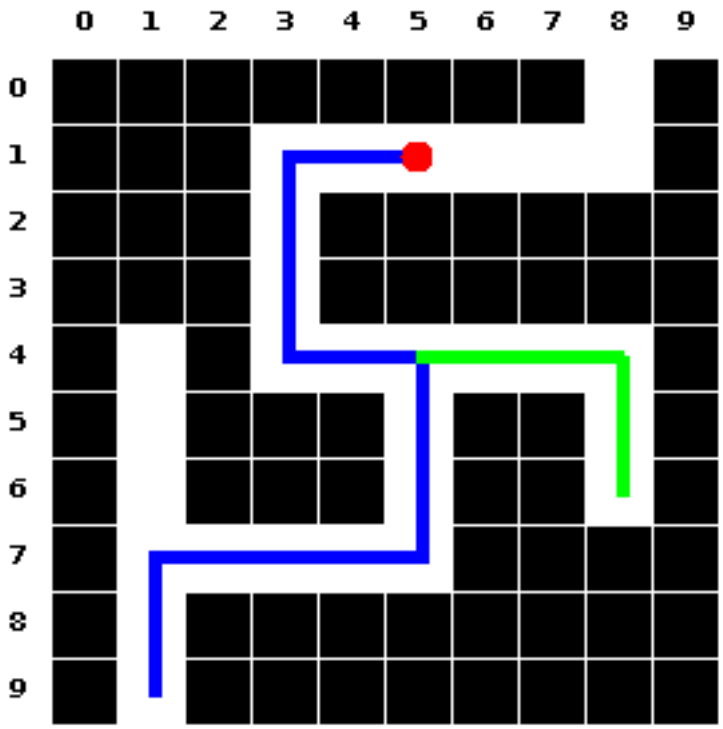

Figure 1: Figure-1

The figure shows 2 separate paths taken by the solver (shown in blue and green) as well as the current position of the solver (shown in red). The paths and markers shown above are very useful in papers describing the progress of a maze-solver during its course of exploring the maze. Below we see a typical random 20x20 maze generated with the MDL generator (and then compiled using the compiler):

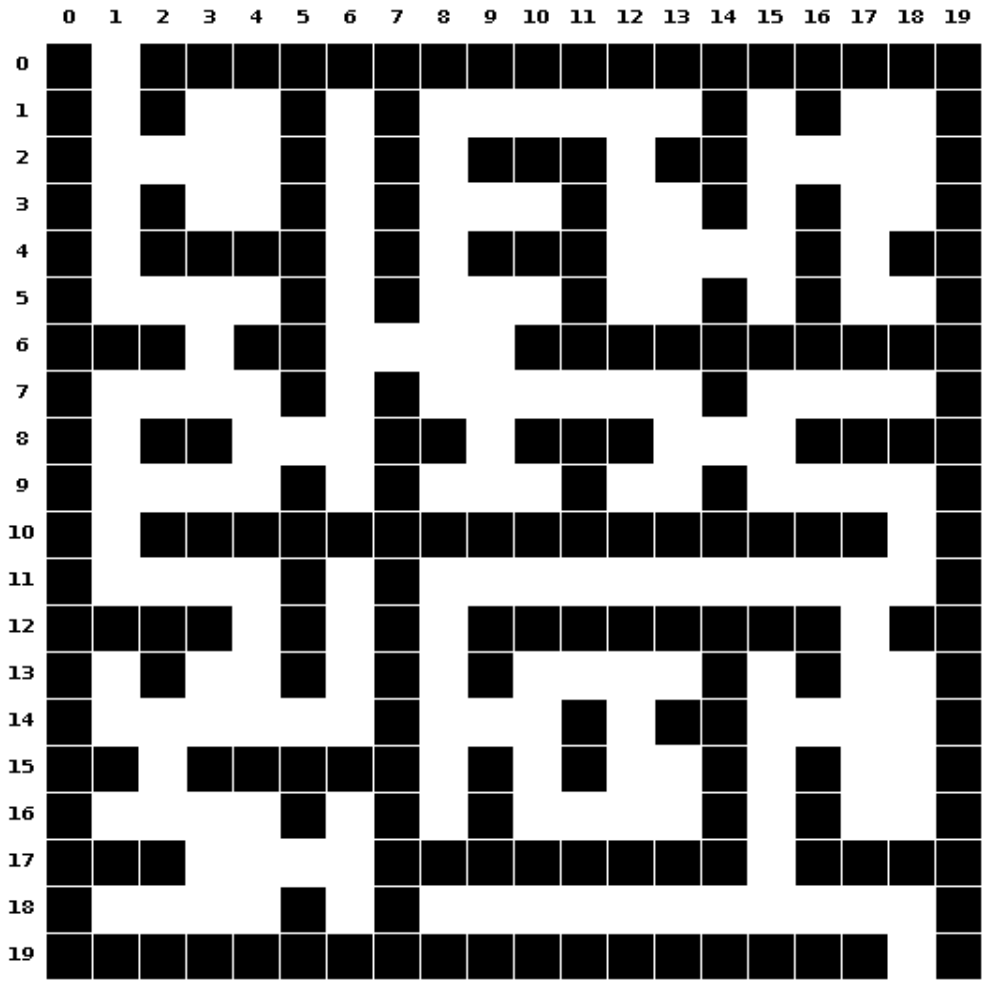

Figure 2: Figure-2

Future Scope

The MDL suite is open source and can be easily extended to develop a GUI application that provides live preview. That will make the maze design process extremely smooth and fast.

Acknowledgements

We acknowledge Susmita Guha for her contributions to this project.

References

Ayaz, H., Allen, S. L., Platek, S. M., & Onaral, B. (2008). Maze suite 1.0: A complete set of tools to prepare, present, and analyze navigational and spatial cognitive neuroscience experiments. *Behavior research methods*, *40*(1), 353–359.

GitHub - oppenheimj / maze-generator. (2018). Retrieved February 6, 2020, from https://github.com/oppenheimj/maze-generator/

Gupta, B., & Sehgal, S. (2014). Survey on techniques used in autonomous maze solving robot. In *2014 5th international conference-confluence the next generation information technology summit (confluence)* (pp. 323–328). IEEE.

History - micromouse online. (2010). Retrieved October 11, 2019, from http://www.micromouseonline.com/micromouse-book/history/

Maze generator. (2019). Retrieved February 6, 2020, from http://www.mazegenerator.net/

Maze generator and solver - wolfram demonstrations project. (2011). Retrieved February 6, 2020, from https://demonstrations.wolfram.com/MazeGeneratorAndSolver/