Visual Parametric Maze Generator DSL ⋆

Corentin Moiny[1]

304, 5e Avenue Mailloux, La Pocatière. Quebec. Canada

University of Montreal[a]

[a] *2900 Edouard Montpetit Blvd, Montreal, Quebec. Canada*

Abstract

(TODO) - This is a test abstract again and again.

Keywords: MDE, Maze, Generator, Parametric, Python, Epsilon, DSL, Java, Visual

2010 MSC: 00-01, 99-00

1. Introduction

A. In the context of our Model-driven Engineering project assignment, I was charged to design a visual DSL to generate parametric mazes using a external Python program that I have also implemented. The goal of this project is to empower parameters understanding with the DSL and than produce probabilistic mazes. With this approach, anyone could generate maze with minimal or no engineering knowledge. Parametric maze generation is not a new concept, our approach was highly inspired by Design-Centric Maze Generation by Paul Hyunjin Kim and al[1]. From this paper we used the maze cells concept where each one of them represent a 3x3 tiles on the maze. We also used the same of types of rates (and more) as in the paper and used them with a probabilist approach.

B. (TODO) - Details of the sections presented

⋆Full source code is available on GitHub.

Email address: corentin.moiny@umontreal.ca (University of Montreal)

[1] 2020

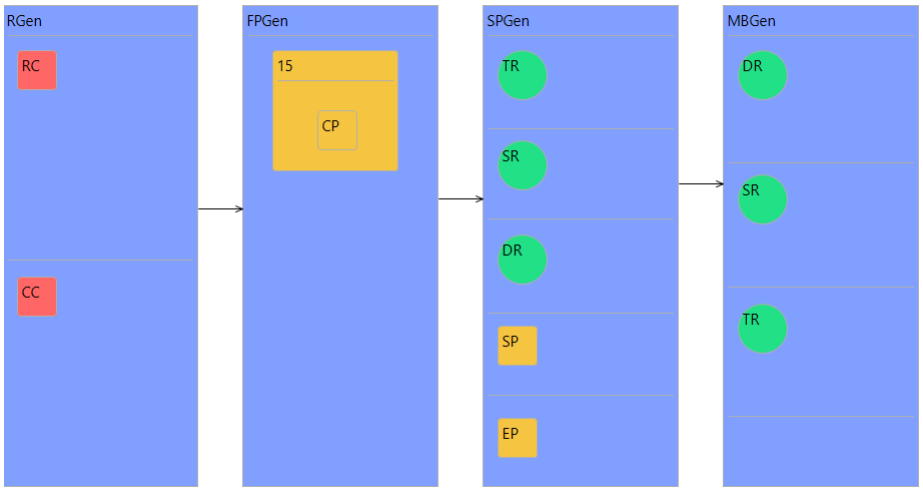

Figure 1: A model instance from the DSL

2. Solution

In this section, I give details on the solution choices used and the purposes behind theses.

2.1. DSL

The domain specific language represent the parameters used to generate the maze. Presented as a visual syntax in Figure 1, it contains four types of generator. From left to right, generators are represented as blue rectangles: (1) *RGen* is the first step of the maze generation, it gives the initial borders of the maze using a row count (RC) and a column count (CC) represented as red squares. (2) *FPGen* inject maze cells in this initial shape to force pattern, it allows users to create drawing in the maze. It is represented as orangish square (*Marked 15 with a CP*) where a point is defined inside of it. In this case we only force a single cell. (3) ... (TODO)

2.2. Generator

3. Evaluation

4. Related Work

5. Conclusion

References

[1] P. H. Kim, J. Grove, S. Wurster, R. Crawfis, Design-centric maze generation, in: Proceedings of the 14th International Conference on the Foundations of Digital Games, FDG '19, Association for Computing Machinery, pp. 1–9. doi:10.1145/3337722.3341854.
URL https://doi.org/10.1145/3337722.3341854