Examination of Representational Expression in Maze Generation Algorithms

Conference Paper · August 2015

DOI: 10.1109/CIG.2015.7317902

3 authors, including:

Aliona Kozlova
Innopolis University
2 PUBLICATIONS **5** CITATIONS
SEE PROFILE

Joseph Alexander Brown
Innopolis University
97 PUBLICATIONS **278** CITATIONS
SEE PROFILE

Examination of Representational Expression in Maze Generation Algorithms

Aliona Kozlova, Joseph Alexander Brown, *IEEE Member*, and Elizabeth Reading
Department of Computer Science, Innopolis University, Kazan, Republic of Tatarstan, Russia 420111
ask.kozlova@gmail.com and jb03hf@gmail.com

***Abstract*—Procedural content generation is widely used in game development, although it does not give an opportunity to generate the whole game level. However, for some game artifacts procedural generation is the preferred way of creation. Mazes are a good example of such artifacts: manual generation of mazes does not provide a variety of combinations and makes games too predictable. This article gives an overview of three basic two-dimensional maze generation algorithms: Depth-first search, Prim's, and Recursive Definition. These algorithms describe three conceptually different approaches to maze generation. The path lengths of the algorithms are compared, and advantages and disadvantages of each algorithm are given.**

I. Introduction

Computational Intelligence algorithms have representation as key factor in their expressive ability. Procedural Content Generation (PCG), see [1], using search based methods has been an emerging topic. Understanding the representation allowed from current maze generation algorithms allows for a understanding in the potential expressive ability and control of the evolution, see [2] for the taxonomic definitions. Statistical analysis of two-dimensional maze creation has been examined for more complex generative methods [3], in terms of path length, but not for some of the base algorithms which can be used for mazes. This is the goal of this demonstration paper.

Mazes are useful in genres of games such as *rougelikes*. Their generation is a challenging task since there are several things that have to be controlled during generation [4]:

- the maze has to have an entrance and exit
- the exit must be reachable from the entrance
- cells that are not included in the path have to be reachable from the entrance
- there must be no cycles in the maze
- any cell in the maze should be reachable
- the amount of walls should be considerable, so that the player will not reach the exit too fast.

That is why by now there is plenty of maze generation algorithms. Three of well-known algorithms are described below.

The remainder of the paper is organized as follows: Section II examines the Depth-first search (DFS), Prim's, and Recursive Definition algorithms. In Section III the algorithms are compared. Finally, Section IV gives the conclusions of the study and advises on a representation to use in future generative methods.

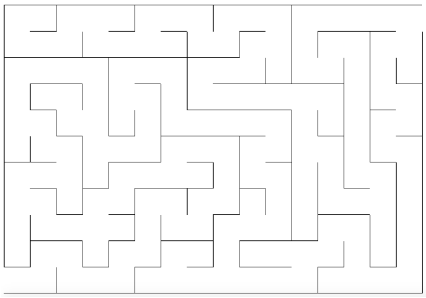

Fig. 1. Map Created by the Depth First Search Algorithm

II. Algorithms

A. Depth First Search Algorithm

The DFS is said to be one of the simplest algorithms to generate maze [5]. Each cell is initialized to have walls on every side. DFS starts with a random cell selection, which is the start, and goes cell by cell breaking walls until it reviews all the cells on the field. The following sequence of actions describes generation algorithm:

1) A random cell is selected to be the starting point (in this paper's implementation the first cell is the starting point). The cell is put into a stack.
2) One random cell from current cell's neighbors is picked.
3) If the picked cell has not been checked by the algorithm yet, the wall between current cell and random cell is broken. The sequence of actions is repeated from the first point, but with the random cell as a starting point.
4) If the picked cell was checked, then we pick another cell from cell's neighborhood and repeat from Step 3.
5) If all the current cell's neighbors were checked, then we go to the previous cell and make it current, then proceed from Step 2.
6) The algorithm stops when all the cells are checked.

The sample output from this algorithm is depicted on 1.

B. Prim's Algorithm

Prim's algorithm [6], starting with all cells initialized as walls, takes one random cell and detects its neighbors (frontiers), one of which is randomly added to the maze. When the cell is added to the maze, the new frontiers are detected. The algorithm goes on until all the cells are in the maze. So the sequence of the actions is the following:

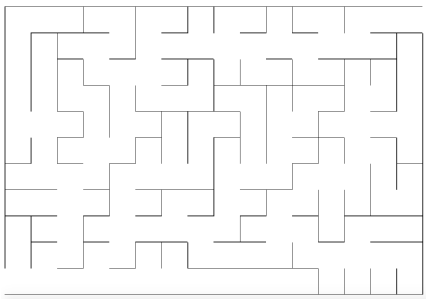

Fig. 2. Map Created by the Prim's Algorithm

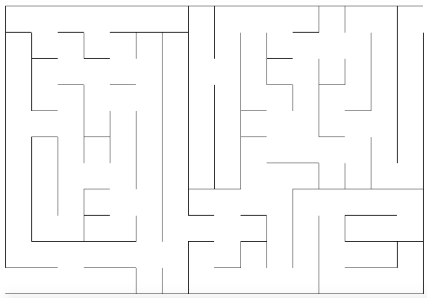

Fig. 3. Map Created by the Recursive Definition Algorithm

1) Select random cell.
2) Detect its neighbors, which are in different set.
3) Choose random neighbor. If it is not in the set, delete the wall between current cell and the neighbor.
4) Go to step one (but now the random cell is selected among cells in the maze).
5) Repeat until all the cells are in the maze.

The sample output from this algorithm is depicted on Figure 2.

C. Recursive Division Algorithm

Recursive division algorithm[7] starts with the grid with no walls, then randomly divides the grid into two parts and puts a random pass in the wall. The algorithm is the following [8]:

1) Divide the grid into two parts by the wall. Put a random pass between two parts.
2) Select the smaller part of the grid to work with. The other is put to the stack.
3) Go to step one and repeat actions with selected smaller part.
4) Proceed until the width or height of one of the parts is one cell.

The sample output from this algorithm is depicted on Figure 3.

III. Comparing Algorithms

Each of the three algorithms were run 100 times to produce a 16x11 size maze as an example of their generation. This size was chosen in order to meet with an iOS development aspect ratio. From the figures we can see that different algorithms give mazes that typically look different. Depth first search algorithm produces mazes that have long and winding passes, which seems to be a good feature for a maze [9]. Mazes generated by Prim's algorithm have shorter passes, players have less chance to get lost in them in comparison with previous algorithm. Mazes generated recursively dividing the grid tend to have long straight walls, which makes it easier to pass through them. The comparison of the algorithms is presented in Table I.

TABLE I
Comparison between the three algorithms in path length on a 16x11 map

Algorithm	Max	Min	Mean	StdDev
Depth First Search	126	40	79.57	20.01
Prim's	34	24	27.78	1.97
Recursive	106	26	61.29	19.81

Using a pairwise two tailed t-test as a comparison method between means, it was found that both DFS and the Recursive algorithms provided longer path lengths than Prim's algorithm ($P < 0.0001$) and that DFS provided longer paths than the Recursive algorithm ($P < 0.0001$). Therefore, representations which focus on DFS methods should be used in practice as the path lengths provided will be maximal. Further, DFS gives a longer minimum path length than Prim's maximal path length.

IV. Conclusions

This study advises to use the DFS algorithm as a base for new maze representations. This algorithm meets with the requirements for generating long maze pathways. DFS as a maze generator is only able to make trees - which would allow for fast applications of a number of existing algorithms used in games design such as A* to examine the graph.

References

[1] Julian Togelius, Emil Kastbjerg, David Schedl, and Georgios N. Yannakakis, "What is procedural content generation?: Mario on the borderline", in *Proceedings of the 2nd International Workshop on Procedural Content Generation in Games*, New York, NY, USA, 2011, PCGames '11, pp. 3:1–3:6, ACM.

[2] J. Togelius, G.N. Yannakakis, K.O. Stanley, and C. Browne, "Search-based procedural content generation: A taxonomy and survey", *Computational Intelligence and AI in Games, IEEE Transactions on*, vol. 3, no. 3, pp. 172–186, Sept 2011.

[3] C. McGuinness, "Statistical analyses of representation choice in level generation", in *2012 IEEE Conference onComputational Intelligence and Games (CIG)*, Sept 2012, pp. 312–319.

[4] Walter D. Pullen, "Maze classification", June 2015, http://www.astrolog.org/labyrnth/algrithm.htm.

[5] Ki Sang Lee, "Maze-generating algorithms", in *Physical Mathematics Conference*, 2009.

[6] Jamis Buck, "Maze generation: Prim's algorithm", January 2011, http://weblog.jamisbuck.org/2011/1/10/maze-generation-prim-s-algorithm.

[7] Jamis Buck, "Maze generation: Recursive division", January 2011, http://weblog.jamisbuck.org/2011/1/12/maze-generation-recursive-division-algorithm.

[8] Emre Zorlu, "Perfect maze creation & path finding algorithms", March 2012, http://emrezorlu.com/2012/03/20/maze-creation-solving.

[9] D. Ashlock, C. Lee, and C. McGuinness, "Search-based procedural generation of maze-like levels", *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, no. 3, pp. 260–273, Sept 2011.