

Procedural Content Generation for Room Maze

Hongqiao Zhang¹, Xiaohua Zeng¹, Lingfei Duan¹

1. College of Electronic Information and Electrical Engineering, Xiangnan University
Chenzhou, China
7223387zhq@163.com

Abstract—Procedural content generation (PCG) has recently become one of the most popular topics in computational intelligence and video games' content research. This paper has proposed an implementation method for procedural content generation of room maze. Firstly, using flood-fill algorithm to make the random rooms' center, inner floors and boundary. Then tree traversal algorithms which are Depth First Search (DFS) and Breadth First Search (BFS), used to build the relation table of rooms and get the door position between different rooms. Finally, this paper has illustrated the different characteristics of DFS and BFS with different relation information during generating the room maze. This method of procedural content generation can be used to make auto dungeon map for video games or provide plenty of connected graphs for the AI learning research.

Keywords—procedural content generation; maze generation; flood-fill algorithm; tree traversal algorithm; DFS; BFS

I. INTRODUCTION

Procedural content generation is a way to generate content interacting with human automatically. It aims to build endless possibilities of artistic content which can reduce a lot of manual configuration [1]. In recent years, PCG is used to the video game content like the sandbox games and dungeon games. It also can be utilized for generating the content of AI learning [2].

The search-based procedural content generation (SBPCG) is a usual way to implement the PCG. It uses different kinds of search algorithms to explore and change the content or style of the given space [3]. This paper contributes a kind of SBPCG based on flood-fill algorithm and tree traversal algorithms to build infinite kinds of 2-D room mazes [4-6]. The random content created by this method has controllable information for further using. The outline of this method is as following:

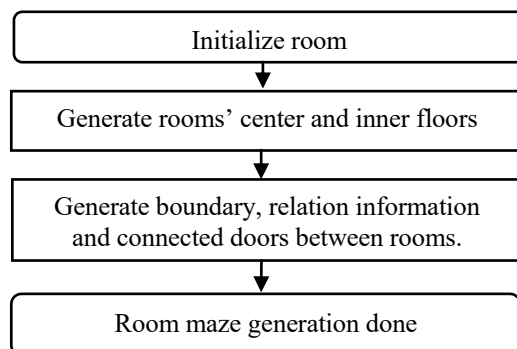


Fig. 1. Outline of the Proposed SBPCG Method

II. RANDOM GENERATION OF ROOMS' AREA

A. Initialization of Room Grid

Two dimensional grid is used to generate the room maze with size of H rows by columns W , and 4 neighborhood is used as the relation between grid cells. Each grid cell has two attributes: room state (STA) and room index (RID). The STA indicates the state of the cell in the room, and the RID indicates which rooms the cell is belong to. The relational values of RID and STA are listed here:

TABLE I. GRID CELL'S ATTRIBUTES

Grid cell's room state and room index		
STA	RID	Details
Empty	-1	Initial cell state which belongs to nothing. If any empty cell has not yet been processed before stage of boundary generation, it will change to the boundary state with $RID=-1$.
Key	-1	Initial cell state which belongs to nothing. These cells have chance to become Center cells.
Center	≥ 1	First cell owned by specific room.
Inner	≥ 1	Room's floors which can be passed through.
Boundary	-1 or values ≥ 1	Blocked cells between rooms. Its RID is a data sets of its 4 inner state neighborhood cells' RID . In other situation, RID is -1.
Door	values ≥ 1	Cells which transformed from Boundary state are used to connect different rooms.

This paper initializes all the grid cells' RID to -1. The cell with the position of odd row and odd column sets the STA to Key, other cells' state are Empty. Fig. 2 illustrates an initialized 7 by 7 grid.

		Empty Cell Key Cell					
col	0	1	2	3	4	5	6
row 0	-1	-1	-1	-1	-1	-1	-1
1	-1	-1	-1	-1	-1	-1	-1
2	-1	-1	-1	-1	-1	-1	-1
3	-1	-1	-1	-1	-1	-1	-1
4	-1	-1	-1	-1	-1	-1	-1
5	-1	-1	-1	-1	-1	-1	-1
6	-1	-1	-1	-1	-1	-1	-1

Fig. 2. Initialization of the 7 by 7 Grid

B. Inner Room Generation based on Flood-fill

Firstly, this paper selects one of the Key cells randomly as current room's Center cell, marking the *RID* as M ($M \geq 1$) and changing the *STA* to Center. Note that attributes of each cell with state Empty or Key can be changed by only once during this stage.

Secondly, around the Center cell, scanning cells of top-left, top-right, bottom-left and bottom-right rectangle sub-areas as current room's potential area, each sub-area has different width sW and height sH , the sW and sH are random odd numbers and ignore the position out of the grid.

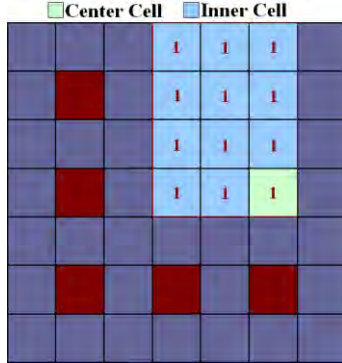


Fig. 3. 5 by 3 top-left Sub-area of Room1

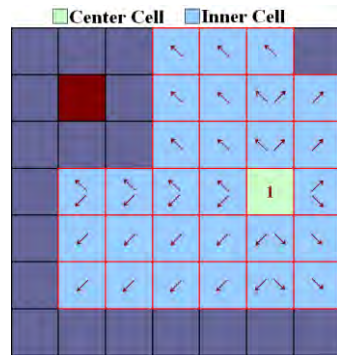


Fig. 4. Room1's Random Potential Area

Considering the cells' connectivity of each room, this paper uses the flood-fill algorithm to fill the potential room area, starting from the position of Center cell. In other words, sometimes a small part of cells in the room's potential area are isolated with the Center cell and should not be contained to current room.

The *RID* of each flood-fill scanned Empty or Key cell will change to the same *RID* of room's Center cell, and the *STA* should be set to Inner at the same time. The procedure of the flood-fill algorithm is shown as Fig. 5.

When starting to generate another new room, we should skip the cells whose *RID* is not -1. The process of this stage is illustrated as Fig. 6.

After this stage, the cells' state of grid should be Empty, Center or Inner as shown in Fig. 7.

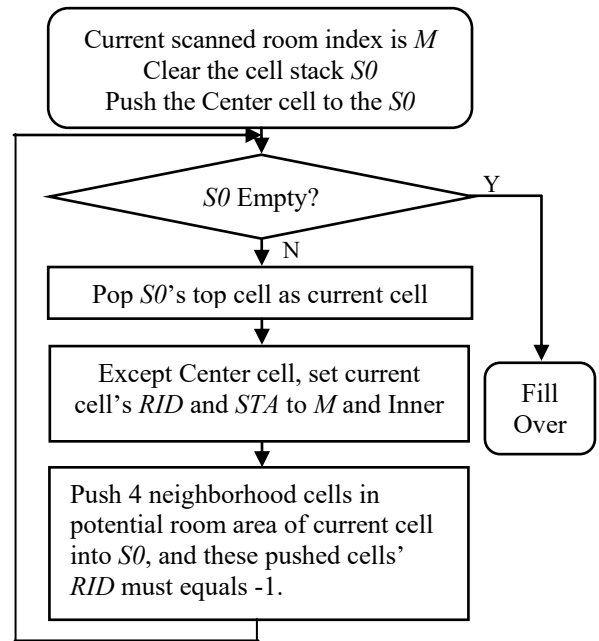


Fig. 5. Process of Single Room with Flood-fill Algorithm

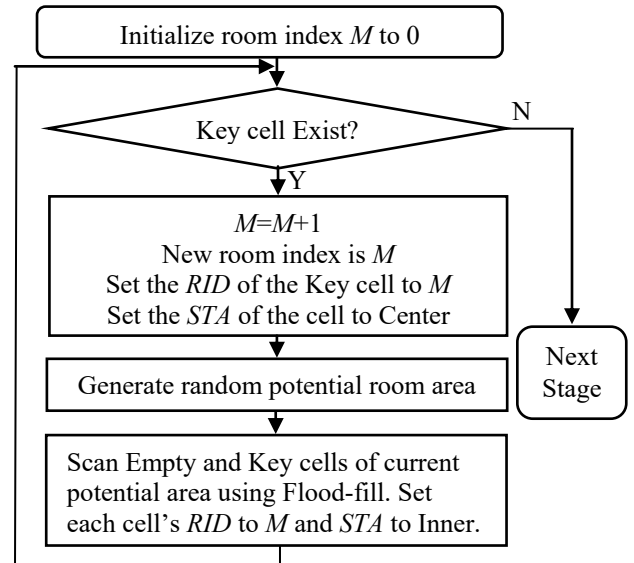


Fig. 6. Generation of Random Rooms' Area

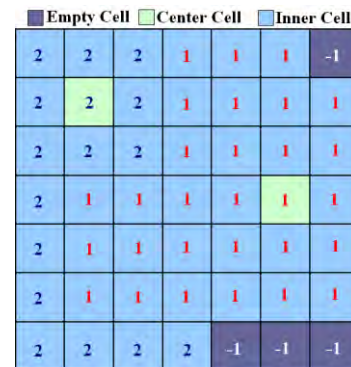


Fig. 7. Random Rooms' Area of 7 by 7 Grid

C. The Generation of boundary

In this part, we scan whole grid again. If the *RID* of current cell is smaller than any cell of its 4 neighborhood, or current cell's state is Empty, current cell's *STA* will recorded as Boundary. This condition will ensure to generate the single-layer boundary. For each boundary cell, there are N different *RIDs* of its four neighborhood Inner cells. If $N > 0$, the *RID* of the boundary cell is a data sets of its four neighborhood Inner cells, otherwise, its *RID* marks as -1. Following Fig. 8 shows the boundary generation result based on the Fig. 7's grid.

Boundary Cell							
2	2	2	(1,2)	1	1	1	(1)
2	2	2	(1,2)	1	1	1	1
2	2	2	(1,2)	1	1	1	1
2	(2)	(1,2)	1	1	1	1	1
2	(1,2)	1	1	1	1	1	1
2	(2)	(1,2)	(1,2)	1	1	1	1
2	2	2	2	(1,2)	(1)	(1)	(1)

Fig. 8. Boundary Generation

III. GENERATION OF ROOMS' RELATION

A. Room Relation Table

The boundary cell has the basic relation information between different rooms. With this stage this paper regenerates the rooms in an 11 by 11 grid shown as Fig. 9 to illustrate the procedure of room relation generation.

Col	0	1	2	3	4	5	6	7	8	9	10
Row	0				(2,5)	(2)	(-1)		(3)		
1		5		(2,5)							
2											
3			(4,5)	(2)		2	(2,3)		3		
4											
5		4		(2,4)		(1,2)	(1)	(1,3)			
6					(1,2,4)					(1,3)	
7			(4,6)		(1)		1				(1)
8											
9		6			(1,6)						
10					(6)		(1)				(-1)

Fig. 9. 11 by 11 Grid with Generated Rooms' Area

This paper makes the room relation table of the room grid with boundary cells' position. The table lists the elements' position [row, column] of boundary cells without the repetitive room relation.

The corresponding Table 2 is created by using the grid boundary information based on Fig. 9. With this table, we can use the boundary cells with $N \geq 2$ or just $N=2$ to make the room tree traversal.

TABLE II. ROOM RELATION OF FIG. 9'S GRID

Source/ Destination Room Index	Destination/ Source Room Index	Room Relation Table		
		Boundary cells' Position [row,col]		
		N=2	N=3	N=4
1	2	[5,5],[5,6]	[6,4]	-
	3	[5,8],[5,9],[6,10]	-	-
	4	-	[6,4]	-
	6	[8,4],[9,4]	-	-
2	3	[1,7],[2,7],[3,7],[4,7]	-	-
	4	-	[6,4]	-
	5	[0,4],[1,3],[2,3]	-	-
4	5	[3,0],[3,1],[3,2]	-	-
	6	[7,0],[7,1],[7,2],[7,3]	-	-

B. Room Relation based on Tree Traversal

This paper builds the rooms' relation tree by two kinds of search algorithms: DFS and BFS based on previous relation table. During the process, each room is accessed only by once. The procedure of DFS is shown as following figure:

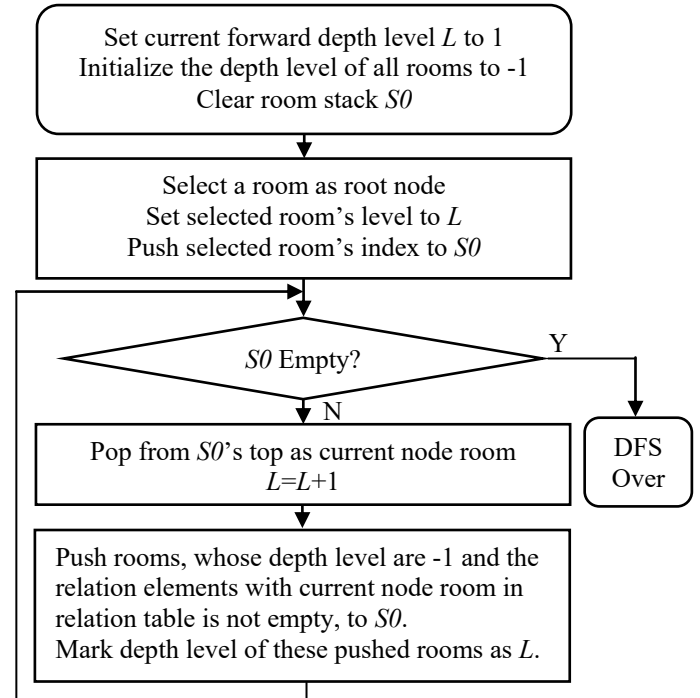


Fig. 10. Procedure of Rooms' DFS

If selecting the room3 as the root node, the DFS trees with different N will look like Fig. 11 and Fig. 12. These figures show that the rooms' tree would be more complex with more kinds of boundary cells.

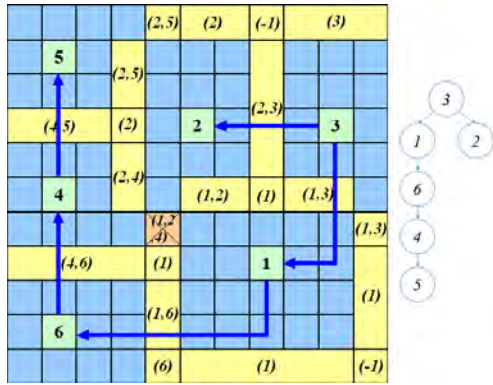


Fig. 11. DFS for Room Relation Tree ($N=2$)

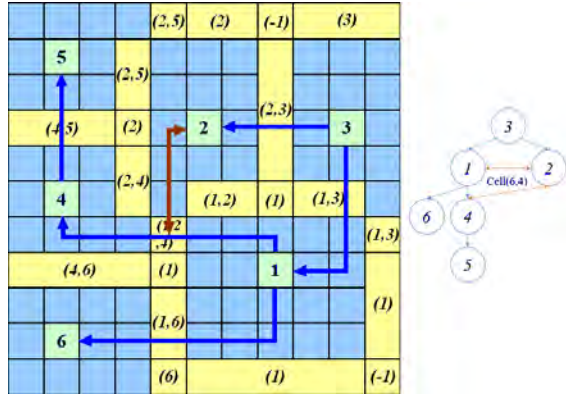


Fig. 12. DFS for Room Relation Tree ($N \geq 2$)

The procedure of BFS is shown as following figure:

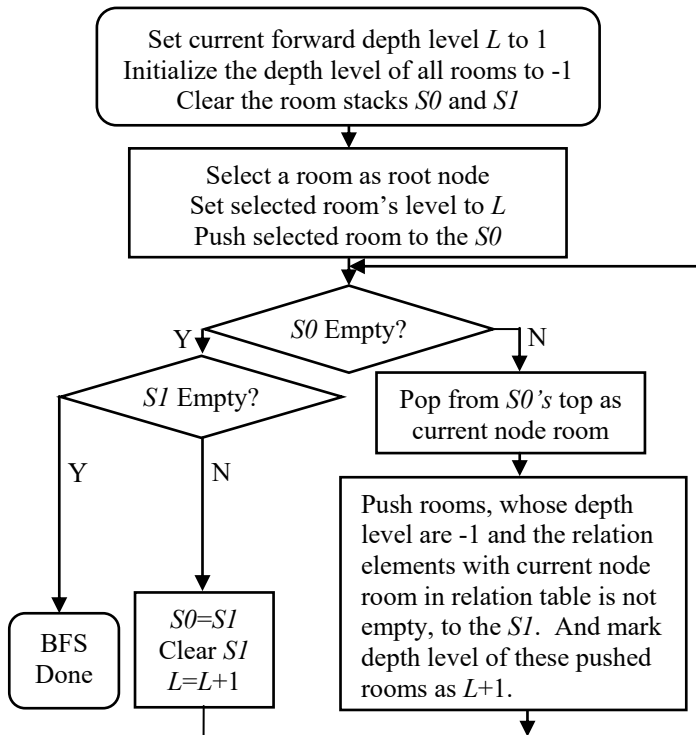


Fig. 13. Procedure of Rooms' BFS

The BFS trees with different N look like Fig. 14 and Fig. 15 when select room3 as the root:

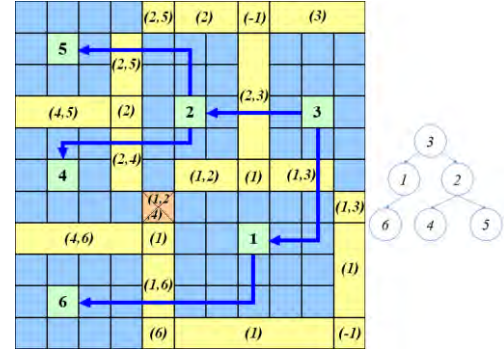


Fig. 14. BFS for Room Relation Tree ($N=2$)

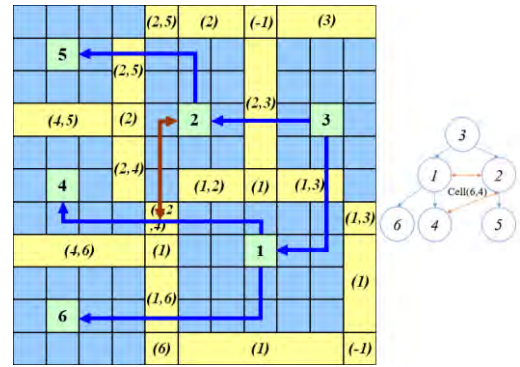


Fig. 15. BFS for Room Relation Tree ($N \geq 2$)

From the illustrated figures, we can find that DFS has more depth levels than BFS. On the contrary, BFS has more room-crossed possibilities than DFS. And the relation table with condition $N \geq 2$ is more complex than condition $N=2$, or we can say the latter condition is more controllable than the former.

C. Generation of Door Cells

This paper uses relation table with condition $N=2$ to build the room tree traversal and select the Door cells from boundary cells between each pair of rooms. Here we randomly pick up a cell of position $[i,j]$ from relation table's element between different two rooms, and change the cell state of corresponding position to Door. After this step, we have generated the latest random room maze shown as Fig. 16:

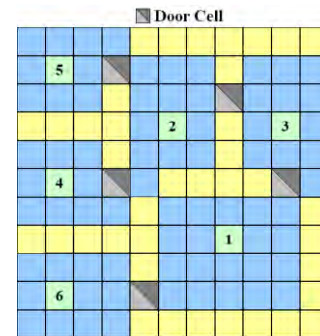


Fig. 16. 11 by 11 Room Maze with BFS ($N=2$)

Fig. 17 and Fig. 18 illustrate the random room maze generation effect with different grid size implemented by the PCG method in this paper.

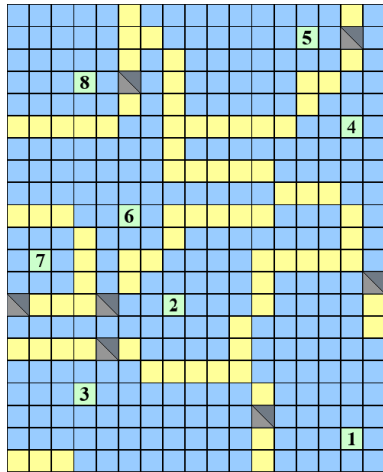


Fig. 17. 21 by 17 Room Maze with BFS

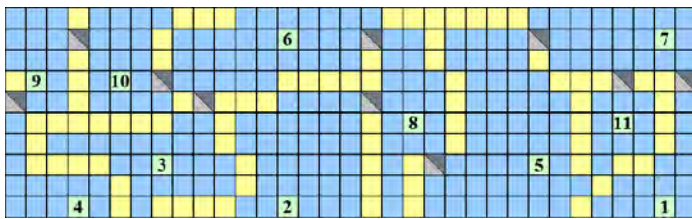


Fig. 18. 10 by 33 Room Maze with DFS

IV. CONCLUSION

This paper has proposed a PCG method to generate the two-dimensional grid based room maze which has randomized structure and controllable room relation. Based on the specific

position to get the rooms' center, we use flood-fill algorithm to generate different rooms' inner floors and boundary cells. During next stage, boundary information is utilized to get the room relation table. With different types of elements in the relation table, DFS and BFS are used to build the room relation tree and make the door between rooms. Meanwhile this paper has illustrated the different characteristics of DFS and BFS under different relation information. This work of procedural content generation can be used to make auto dungeon map for video games or provide plenty of connected graphs for the AI learning research.

ACKNOWLEDGMENT

This work was supported by the Xiangnan University School Level Scientific Research Project in 2017 under Grant No. 2017XJ22.

REFERENCES

- [1] Flora Amato and Francesco Moscato, "Formal Procedural Content Generation in Games Driven by Social Analyses," 2017 31st International Conference on Advanced Information Networking and Applications Workshops, 2017, pp. 674–679.
- [2] Jonathan Roberts and Ke Chen, "Learning-Based Procedural Content Generation," IEEE Transactions on Computational Intelligence and AI in Games, 2015, vol. 3, issue 1, pp. 88–101.
- [3] Miguel Frade, F Fernandez de Vega, Carlos Cotta, "Aesthetic Terrain Programs database for creativity assessment," 2012 IEEE Conference on Computational Intelligence and Games, 2012, pp. 350–354.
- [4] Zhuang Cai, Lu Ye and Ang Yang, "FloodFill Maze Solving with Expected Toll of Penetrating Unknown Walls for Micromouse," 2012 IEEE 14th International Conference on High Performance Computing and Communications, 2012, pp. 1428–1433.
- [5] Balram Swami, Ravindar Singh, "Performance Analysis of DFS based Ordered Walk Learning Routing Protocol in MANET," 2015 International Conference on Green Computing and Internet of Things, 2015, pp. 195–198.
- [6] Ishwar Baidari, Ajith Hanagawadimath, "Traversing directed cyclic and acyclic graphs using modified BFS algorithm," 2014 Science and Information Conference, 2014, pp. 175–181.