A Maze Game on Android Using Growing Tree Method

IOP Conf. Series: Journal of Physics: Conf. Series **953** (2018) 012148 doi:10.1088/1742-6596/953/1/012148

A Maze Game on Android Using Growing Tree Method

Y F Hendrawan

Department of Informatics Engineering, University of Trunojoyo Madura, Jl. Raya Telang, Kamal, Bangkalan, Madura 69162 Indonesia

Email: yonathan.hendrawan@trunojoyo.ac.id

Abstract. A maze is a type of puzzle games where a player moves in complex and branched passages to find a particular target or location. One method to create a maze is the Growing Tree method. The method creates a tree that has branches which are the paths of a maze. This research explored three types of Growing Tree method implementations for maze generation on Android mobile devices. The layouts produced could be played in first and third-person perspectives. The experiment results showed that it took 17.3 seconds on average to generate 20 cells x 20 cells dynamic maze layouts.

1. Introduction

A maze is a kind of games where a player moves in pathways with many branches to find a way out/certain targets [1]. Maze and labyrinth are two different things. The labyrinth is not as complicated maze: labyrinth's path that a player can tread has no branching at all. But usually, people consider maze and labyrinth are the same [2].

Of the diverse types of mazes, the one commonly encountered by people is the perfect maze type. Perfect here means that the maze does not have not only loops/circular paths, but also insulated parts which no other parts can access to. Figure 1 shows an example of a perfect maze layout.

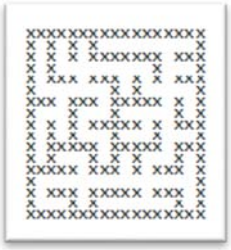

Figure 1. Maze in text mode 8 x 8 cell.

Maze-making process is the process of designing the position of paths and walls in the maze. There are many methods/algorithms to generate a maze, but essentially these methods can be categorized into two groups: step-by-step creating a path in the maze that is wholly covered by walls or building walls on the empty field/space. In both methods, random values are used in determining the next path or wall which is going to be built. Because of the randomness and the steps taken in the methods, computers are an appropriate tool for building mazes, since computers not only have the abilities in creating and using random values easily but also can execute certain steps of algorithm repeatedly without being exhausted.

There are many methods to create a perfect maze layout. Depth First Search, Prim, Eller, Kruskal, and Aldous-Broder are some of them. Each has advantages and disadvantages [3]. Many maze games have been created and explored: building dynamic maze in applet environment [4], making a social maze game to encourage training for multiple sclerosis patients [5], creating a maze based mobile application for learning foreign language [6], using 3D OpenGL ES gravity maze game to develop and apply built-in sensors in Android mobile phones [7], constructing maze games using an Android game framework as part of teaching programming for college students [8], and many other research that have a similar topic.

Mobile devices have been growing rapidly during the last decade. Computational power of mobile devices has increased considerably. This capability is needed by the operating system and various applications (e.g. 3D real-time games) to provide a rich and attractive user experience [9]. Android, as one of many operating systems used in mobile devices, has diverse game applications built on it. Several of them explore the serious side of game such as first aid education for Autism Spectrum Disorder people [10], and teaching forestry lessons in a quiz game [11].

In gaming, maze-making process is part of Procedural Content Generation field, since it creates the level/world to be played on the fly and in random manner by following a procedure/algorithm with no human intervention [12]. One of the advantages of this technique, compared to the manual making technique, is that the game makers do not need to create detailed assets for each part of the game. Game manufacturers just need to write a procedure for making the level once, and then the procedure is the one that will be making different kind of levels in relatively unlimited quantities. Each level generated can also be made such that the difficulty is different: either increasing or decreasing.

In this study, an app/maze game that can generate maze layouts dynamically on Android mobile devices was built. What is meant by the dynamic here is that the maze will be re-created each time the player enters the game levels. The process of making paths and walls of the maze occurs randomly, so the chance of a player getting the exact same maze in succession is small. The method used is the Growing Tree method. This application is also used to measure the level of performance the mobile device has in solving problems.

2. Method

This study uses the Growing Tree method, a list data structure, and Android as the operating system. The game system itself is a puzzle based game.

2.1 Growing Tree Method

Growing Tree method is a method that creates paths which resemble tree branches using a list as the main data structure [5]. Growing Tree algorithm is shown in figure 2.

```
Initialization: pick one cell as the current cell. Put it into a list.
While list is not empty:
    If there is (an) unvisited neighbour cell(s) to the current cell:
       - Randomly crave a path to the cell's unvisited neighbour cells:
       - Put the neighbour cell into the list.
       - Make the neighbour cell to be the current cell.
    Else (all neighbour cells are already visited):
       - Remove current cell from the list.
       - Pick a cell from the list and make it current.
```

Figure 2. Growing Tree Algorithm.

If the list has not been empty, the procedure will still be working by dismantling walls, swapping current cell, adding and removing cells to/from the list. Figure 4 shows the process of making a Growing Tree maze.

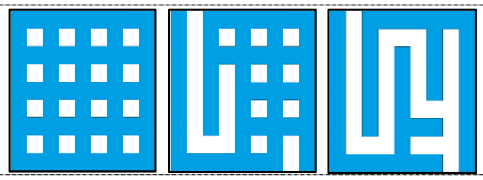

Figure 3. Layout changes in the process of making 4 x 4 cell maze. Left: initial layout, middle: after some walls have been dismantled, right: final layout.

2.2 Three sub-types

There are three ways in picking a cell from the list:

1. Last added cell. The procedure picks the cells in the list using the Last In, First Out (LIFO) technique. The algorithm operates in the same way as recursive backtracker / Depth Search First algorithm. Layouts shown in figure 4 are created using this method.
2. First added cell. The procedure selects the cells in the list using the First In, First Out (FIFO) method.
3. Random. The procedure selects the cells in the list in a random manner.

Figure 4 shows layouts made by using three methods above.

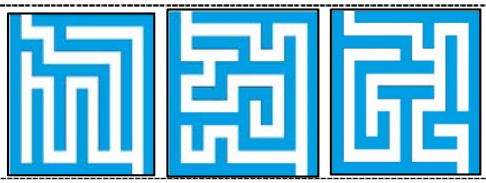

Figure 4. Three sub type layouts for 6x6 cell maze. Left: Last added cell, middle: First added cell, right: Random.

2.3 Gameplay

The shape used in the game is a square, where the number of cells on the vertical side of the maze is equal to the number of cells on the horizontal side.

There are 19 difficulty levels in the game. For each level increase, the number of cells on the vertical and horizontal sides each is increased by one. Starting with 2 x 2 cells for the first level, the difficulty increases to 3 x 3 for the second level, then increases up to the last level with 20 x 20 cells.

After creating the maze, the application displays an avatar on the top left side of the maze (first row, first column). The task of the player is to navigate the avatar towards the exit located on the lower right side of the maze. As a characteristic of puzzle games, the player must solve puzzles presented: finding the right path among the many possible paths.

If the player successfully completes a level, the application displays a winning sign. After that, the player returns to the main screen to select other difficulty levels.

2.4 Mobile Device

This application is built on Android operating system. To display the maze images, the application uses OpenGL. There are two views:

1. Top view, where the player can see all parts of the maze. The player navigates the avatar in third person perspective.
2. Inside the maze, where the player is in the maze's path. The player navigates the avatar in first person perspective.

The game uses a touchscreen interface as its main controller. The player issues commands to the application by using the buttons. There are various buttons each with different function. For examples: to get into a certain level, player must press up or down arrows that set the level number. In the game,

navigation buttons use a familiar format of left, right, up, and down arrows located around the edge of the screen.

3. Results

When the application is started, it displays the main menu screen. From this screen, the player can select one of the available maze levels. There is also an option button to select which of the sub type method will be used to generate the layout.

By default, the level starts from 4x4 cell maze. Figure eight shows the level number located in the middle of the up and down arrow buttons. Pressing the up-arrow button increases the level, and vice versa for the down arrow button. The "Last" word is an option button that shows what sub type of the Growing Tree is used by the application in creating the maze layout. Last represents LIFO method. Pressing this button changes the label to "First", which denotes FIFO method. Pressing it again alters the label to "Random" method. Pressing the "Play" button takes the player to the game screen. On this screen, first the application presents the maze-making process as shown in Figure 4. Each process that changes the shape of the maze is shown one by one on the screen. After that, the avatar, in the form of a space ship, is displayed along the entry (top left) and exit (bottom right) of the maze. The player moves the avatar using the navigation buttons on the right and left sides of the screen. Figure 5 shows the game screen.

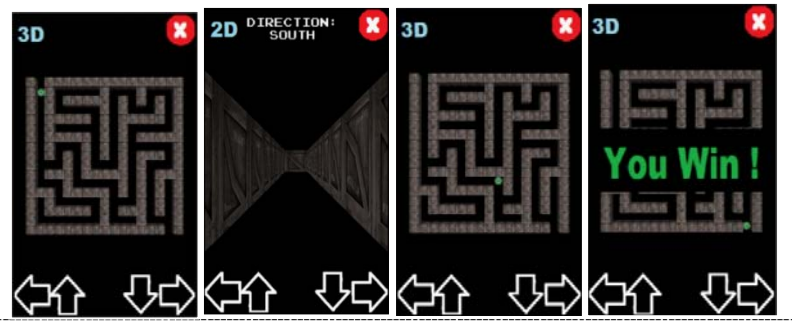

Figure 5. Game screen for 10x10 cell maze. Left to right: the game finished creating the layout and the avatar was placed on top-left cell; first person perspective; the avatar moved around; the avatar reached the exit on bottom-right cell.

The player can choose whichever perspective he or she wants to play the game. The controller adjusts the buttons' function automatically. To switch between perspectives, the player just need to press the 3D/2D button located on the top-left of the screen. In the first-person perspective, the application displays the direction in which the avatar is currently facing on the top-centre of the screen. This information helps in reducing the player's confusion in moving the avatar inside the maze.

Each time the player selects a level, the application creates a new maze pattern. Because of the random nature of the algorithm, it is less likely that the user encounters the same maze layout sequentially. Figure 6 shows several examples of maze layouts generated by the application.

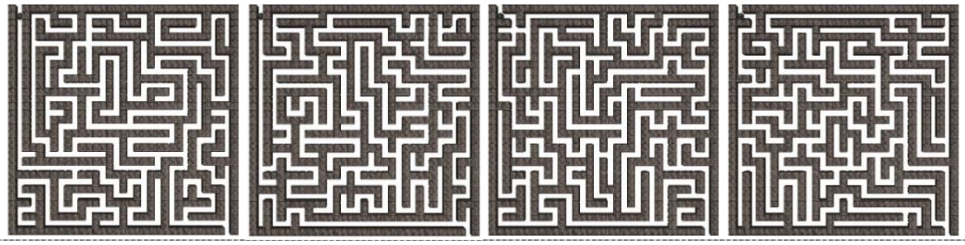

Figure 6. Several 16x16 cell maze layouts. None of them are the same.

Table 1. Average time needed to create a maze.

Level	Cell number	Wall number	Rounded average (seconds)
4 x 4	16	24	0.5
8 x 8	64	112	2.4
12 x 12	144	264	4.3
16 x 16	256	480	8.9
20 x 20	400	760	17.3

For speed testing, this study used Xiaomi Note 2 which has specifications as follows: Mediatek MT6795 Helio X10 as chipset, Octa-core 2.0 GHz Cortex-A53 as CPU, PowerVR G6200 as GPU, Memory 16 GB, 2 GB RAM, 1080 x 1920 pixels screen display.

Speed testing was done by recording the time it takes to make the maze. Recording is done five times for each level. Table 1 shows the average values of the recording results.

In general, the application produces maze layouts quickly. Players do not have to wait for a long time after he or she presses the play button to the point of being able to start moving the avatar.

Figure 9 displays the data in Table 1 in graphical form. The x-axis represents the number of walls (column 3 of table 1) in logarithmic scale format. It also shows a trend line that corresponds to the existing data: $y = -2\mathrm{E}^{-10}x^4 + 3\mathrm{E}^{-7}x^3 - 0.0001x^2 + 0.0358x - 0.2834$ with R-squared value equals 1. This means that the running time of the application is in polynomial order to the number of the walls.

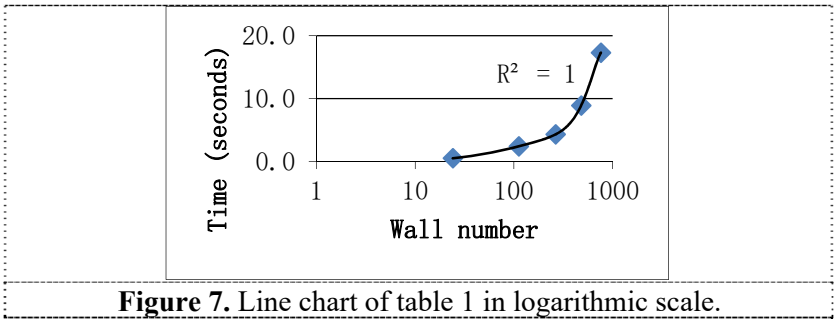

Figure 7. Line chart of table 1 in logarithmic scale.

4. Conclusion and Future Work

The application that was built on Android platform can produce various proper maze layouts dynamically. The maze making process runs reasonably fast as can be seen that to generate 20 cells x 20 cells maze layout, it took 17.3 seconds on average.

From the gaming side, this application can be improved by adding extra challenges such as: putting a time limit for each level and placing bonus items randomly in the maze.

References

[1] Matthews W H 1970 *Mazes and Labyrinths: Their History and Development* (New York: Dover Publications)
[2] Kern H 2000 *Through the labyrinth: designs and meanings over 5000 years* (New York: Prestel)
[3] Pullen W D 2014 *Maze Classification*, http://www.astrolog.org/labyrnth/algrithm.htm accessed 14 June 2017
[4] XUE H Y and ZHAO X Y 2008 Design and Realization of Maze Game Based on Object-oriented *J. Computer and Modernization* **10** 042

[5] De Weyer T, Robert K, Hariandja J R O, Alders G and Coninx K 2012 The social maze: A collaborative game to motivate MS patients for upper limb training *International Conference on Entertainment Computing* (Berlin, Heidelberg: Springer) p 476-479
[6] Todd R W and Tepsuriwong S 2008 Mobile mazes: Investigating a mobile phone game for language learning *J CALL-EJ Online* 10(1) p10-1.
[7] Zhi-An Y and Chun-Miao M 2012 The development and application of sensor based on android *8th International Conference on Information Science and Digital Content Technology (ICIDT)* (Jeju Island: IEEE) p 231-234.
[8] Bayzick J, Askins B, Kalafut S and Spear M 2013 Reading mobile games throughout the curriculum. *Proceeding of the 44th ACM technical symposium on Computer science education* (Denver: ACM) p 209-214.
[9] Zechner M 2012 *Beginning Android Games 2nd* (New York: Apress)
[10] Duveskog M, Laine T H, Arevalo J, Raisanen V, Kirongo B and Orina A May 2013 EntVenture—From binary trees to Kenyan forests: An Android game designed by students. *IST-Africa Conference and Exhibition (IST-Africa)* (IEEE) pp. 1-11.
[11] de Urturi Z S, Zorrilla A M and Zapirain B G July 2011 Serious Game based on first aid education for individuals with Autism Spectrum Disorder (ASD) using android mobile devices. *16th International Conference on Computer Games (CGAMES)* (IEEE) pp. 223-227.
[12] Shaker N, Togelius J, and Nelson M J 2016 *Procedural Content Generation in Games: A Textbook and an Overview of Current Research* (New York: Springer).