The Maze - Help Index

How to play - Instructions

The object of this game is to find your way through the maze from the upper right corner to the lower left corner. Move through the maze using the arrow-keys of your keyboard. Use the CTRL-button in combination with the arrow-keys to jump to the next crossroad.

Definition of keys that can be used with this game:

CTRL-N	Start/generate a new maze
CTRL-O	Open a previously saved maze
CTRL-S	Save the current maze to a bitmap
CTRL-B	Save screenshot into a bitmap (BMP) file (ca. 1.1 MB)
CTRL-R	Restart the Game
CTRL-L	Solve the maze
ESC	Solve the maze
CTRL-X	Exit the program
CTRL-T	Use a tail for walking through the maze
F1	This Helpfile

These commands can also be reached through the menu system of the game.

By clicking on the up/down-buttons the width, height and linesize of the maze can be altered. By clicking on the width or height label, their respective values can be increased by one. By clicking on the size label, the linesize can be increased by five. After you have done this, you must press CTRL-N to generate a maze of the specified size. When the specified maze is too large to fit on the form, the linesize is automatically adjusted, so that the maze fits to the form.

Go back to the Help Index

Implementation of the game

- Source code
- Generating the maze
- Solving the maze

Go back to the Help Index

Source code

The initial purpose of this program for me was to help me learn Borland Delphi 2.0 and many of its aspects. Also, as I am a mathematician, finding an algorithm for generating and solving a maze is fun! For those interested in the Delphi source code of this program, please send a postcard from the city you come from to the following address:

E.W. Hans
Witbreuksweg 393-305
7522 ZA Enschede
The Netherlands

Don't forget to mention your email-address, and your comment to this game. The complete source code of the program will be sent to you by email as quickly as possible.

In case you have any suggestions to this game (e.g. add-ons, bugs, etc.) please email your comment to: e.w.hans@wb.utwente.nl
This email-address cannot be used to obtain the Delphi source code of this program!

Go back to the Help Index

Generating the maze - the algorithm

The maze can be created using Prims algorithm and the so called snake crawler (two different types). Also a unicursal maze can be created. Version one is derived (the algorithm is explained below) from a Prim-maze and version two from the snake crawler.

Prims algorithm:

At the start, the maze is completely filled with a grid. The maze is generated by creating two random trees (i.e. trees in graph theory) in the maze. These two trees start at two random points in the maze. In each iteration each tree is expanded with one node by removing a random line from the maze. Of course only the lines that will expand the trees, and dont make circuits are to be removed. After the maze is completely filled with the trees (note: the trees may not connect!), one final random line is removed to connect the trees. By connecting the trees in this way, the maze is completely filled with a new (larger) tree.

An important characteristic of a tree, is that there always is only one route from point A to B. Consequently, there is only one way between each two points in the maze.

The question may arise why I have created one large tree by connecting two smaller trees, and not just created one large maze from the start. Fact is that generally with my strategy, the maze becomes more difficult to solve. A recent idea of mine is to connect f.e. 4 trees in the following way:
Suppose the trees (t1, t2, .., t4) fill the maze in such a way, that the following trees are adjacent:
t1 & t2
t1 & t4
t1 & t3
t2 & t3
t3 & t4
Suppose the entrance of the maze lies in t2, and the exit in t4. When each tree is represented by a node in a graph, a longest path algorithm can be used to determine the longest path from t2 to t4 (e.g. t2 -> t1 -> t3 -> t4). At this stage, random lines between t2 & t1, t1 & t3, t3 &t4 can be removed to create a maze with the longest path! Got the idea? What do you think about this method? How many trees must be used to make the maze as difficult as possible? Please mail your comments / suggestions to:
e.w.hans@wb.utwente.nl

Snake crawler algorithm:

With this simple algorithm ten random starting points are chosen, from where a snake will grow. Each step the snake will grow until finally the maze is filled. When a snake cant grow (the head of the snake is caught within the snakes body), a random continuing point is chosen somewhere on the side of the snake (the snake will thus branch). By finally connecting the snakes, a maze is generated with very few dead ends. A unicursal maze can be created, with only one snake.

Unicursal mazes:

A unicursal maze is a maze with very few (or no) dead ends. A unicursal maze can easily be derived from another maze by bisecting each passage in the maze (i.e. each passage in the maze is to be split by a line that does not touch a wall of the original maze). This way a unicursal maze can be derived of exactly twice the size of the original maze.

Go back to the Help Index

Solving the maze - the algorithm

The maze is solved using a recursive depth first search method. My experience with this method is that it is very effective. When solving the maze a so called gauge is displayed, that displays the progress of solving. In fact updating this gauge takes about half of the time of solving the maze!

Go back to the Help Index

About the author

This program was made by E.W. Hans in the Netherlands. For all people that enjoyed playing this game, or that are interested in the source code of the game (Delphi), please send a postcard from the city you come from to the following address:

E.W. Hans
Witbreuksweg 393-305
7522 ZA Enschede
The Netherlands

Don't forget to mention your email-address (in case you want to recieve the Delphi source code), and your comment to this game. Comment can also be sent to my email-address:
e.w.hans@wb.utwente.nl

Go back to the Help Index