Shortest Path

Ciprian Habuc

Master student in Computer Science Department,
"Babeş-Bolyai" University, Cluj-Napoca, Romania
chippry@gmail.com

February 20, 2009

1 Introduction

A **maze** is defined as a complex tour puzzle in the form of a complex branching passage through which the solver must find a route.

There are various types of mazes. One of the most interesting types is the *perfect* maze; it has no loops and no cell that can not be reached.

For generating mazes there are many algorithms. The most used are:

- Prim's algorithm
- Kruskal's algoritm
- Recursive division

The most common ways for solving mazes are:

- Wall follower
- Tremaux's algoritm

2 Recursive division

One of the most simple way for generating a perfect maze is the recursive division. It starts with a rectangular space called *chamber*. Then build at random points two perpendicular walls that will divide the original chamber into four smaller chambers separated by the four walls. On three randomly chosed walls pick one cell-wide holes at random points. Continuing recursively this way until each cell represents a chamber a perfect maze will result.

This method is pictured in the following figure:

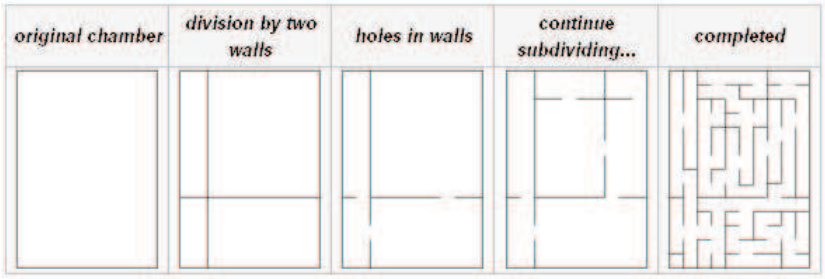

Figure 1: Ilustration of recursive division

The coresponding algorithm will be:

Algorithm 1 recursiveDivision(Chamber)

$A, B, C, D \Leftarrow pickRandomly4PerpendicularWalls$
$pick3HolesOn3OfTheWalls$
$recursiveDivision(A)$
$recursiveDivision(B)$
$recursiveDivision(C)$
$recursiveDivision(D)$

3 Habuc's algorithm

Wouldn't it be useful to have a way for finding the shortest path for a perfect maze since the maze creation?

3.1 Description

The algorithm starts by storing the entrance(initial) point of a rectangular space(*chamber*). Then it picks two perpendicular walls that will produce four new smaller chambers: A, B, C, D. On three of them pick at random points one cell-wide cells. At this point depending on the position of these three points and the entrance and exit of the big chamber there are 28(=4*(1+2+2+2)) posibilities for the shortest path to be built.The rooms that has to be passed while following the shortest path will be recursively processed by this algorithm (storing key points) and the left ones may be processed by the recursive division algorithm. Finally it adds the exit point of the big chamber. Now the solver should join the neighbour points (that have a passage between them and no walls) with lines. This line represents the shortest path.

3.2 Examples

The method can be described by the following algorithm:

Algorithm 2 habuc(Chamber)

shPList.add(Entrance) {add entrance point to the list }
$A, B, C, D \Leftarrow pickRandomly4PerpendicularWalls$
$pick3Holes(x, y, z)On3OfTheWalls$
$listCompulsory \Leftarrow getListOfCompulsoryChambers(x, y, z, Entrance, Exit)$
$listNotCompulsory \Leftarrow getListOfNotCompulsoryChambers(x, y, z, Entrance, Exit)$
for all X chamber in $listCompulsory$ **do**
  habuc(X)
end for
for all X chamber in $listNotCompulsory$ **do**
  recursiveDivision(X)
end for
$shPList.add(Exit)$ {add exit point to shP - list of key points}

Let the images speak:

$secondsubimage$ - let's supose the entrance of the maze is the top-left point and the exit the bottom-right point

$forthsubimage$ - after picking the for walls and the three holes it is obvious that there is no use for a solver to explore room B

$fifthsubimage$ - the built maze and the stored shortest path

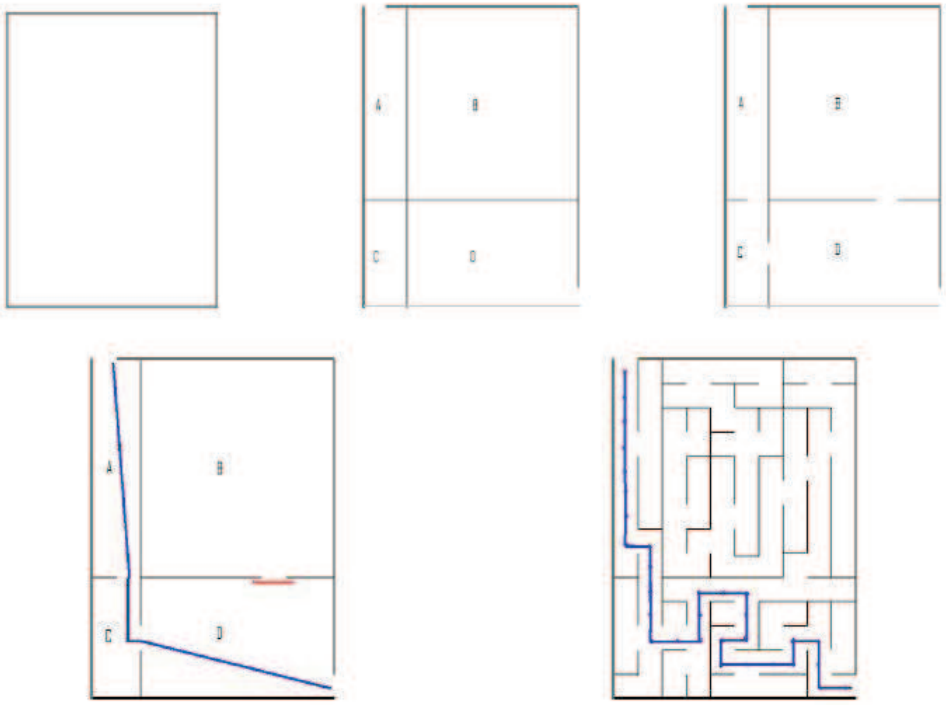

Figure 2: Ilustration of shortest path

Figure 3: Perfect maze and shortest path

4 Conclusions

Advantages

- it creates a perfect maze and stores the shortest path
- shortest path can be used for comparison with results from other algorithms

Disadvantages

- it can be used only for perfect mazes