A-MAZE-ING GAME

Samuele Galbusera - Lorenzo Spada - Amedeo Zappulla

December 13, 2023

DEVELOPER GUIDE.

Contents

1 Introduction

A-MAZE-ING game is a program that solves mazes created from users.
The Breadth-first search, BFS, is the main algorithm behind the project, and is fully implemented without the use of any external library.
The language used to implement the program is DrRacket ASL (Advanced Student).

2 Libraries

2.1 index

The following library was used to implement the program:

- data/queue
- 2htdp/universe
- 2htdp/image

2.2 data/queue

Library required to implement the BFS, is also possible use list instead queue but it would be necessary to create some functions which allow to use list as queue.

2.3 2htdp/universe

Allow to use the big bang function, which run the entire project.

2.4 2htdp/image

Import images and shape used to draw the maze.

3 Main functions

3.1 change-matrix

define (change-matrix matrix cell val) . . .)
Change a singular cell of the matrix with val, the cell to change is cell (cell is a posn, the x is the column and the y is the row). This function is used by the BFS to set a cell to be visited by setting its distance from the starting position. It is also used by handle-mouse, to set the cells of the matrix to walls or to empty cells.

3.2 check-node

(define (check-node matrix lst)...)
Check every adjacent node of the first element in the queue, set their distance and update the queue following the BFS algorithm (that is it adds the cell to the queue if its distance isn't set). This function is used inside the BFS function.

3.3 bfs

(define (bfs matrix) . . .)
Executes the bfs, every time this function is called, an element of the queue has ben pop, this function will be called as long as there are elements in the queue

3.4 onTick

(define (onTick state)...)
Start the BFS at every tick and saves the result on the constant MATRIX

3.5 show-path

(define (show-path state canva)...)
Return the canva with the path showed if a path exist

3.6 find-path

(define (find-path matrix canva pos)...)
Draw a line from the current cell to the next cell in one of the shortest paths (once the bfs is run to find the shortest path you start from the end and move one cell at a time to a cell that has a distance from the start equals to the distance from the start of the current cell minus 1)

3.7 get-cell

(define (get-cell x y) . . .)
It takes as inputs to numbers x y (in our program are always the x and y coordinates of the mouse) and returns the correspondent position on the maze matrix after performing a click or a drag on the canva. This function is very useful for handling the mouse events, because it tells the program what element inside the matrix must be changed in order to maybe place or delete a wall in the maze.

3.8 handle-mouse

(define (handle-mouse ms x-mouse y-mouse mouse-event)...)
Takes as inputs the maze-state (appstate), the x and y coordinates of the mouse and the mouse event, which is basically click or drag the mouse.
The handle mouse allow to:

- place walls by clicking on a free space or by dragging.
- move the starting and the ending point of the maze
- delete walls by clicking on wall or by dragging on it.

Obviously this action edit the matrix as show below

Action	State	Matrix Value
Place Wall	State -1	-1
Delete Wall	State 0	0
Entry point	State 1	no value, it's used a posn
Exit point	State 2	no value, it's used a posn

Table 1: Table with Actions, States, and Matrix Values

The idea behind it is basically that every mouse event must not conflict with other mouse events (or the previous maze state), for example the user cannot place a wall at the same position of another wall or of the begin or the end of the maze, or every action they perform outside the maze will result in actually nothing.

3.9 handle-key

(define (handle-key ms key) ...)
Takes as inputs a maze state and a keyboard event

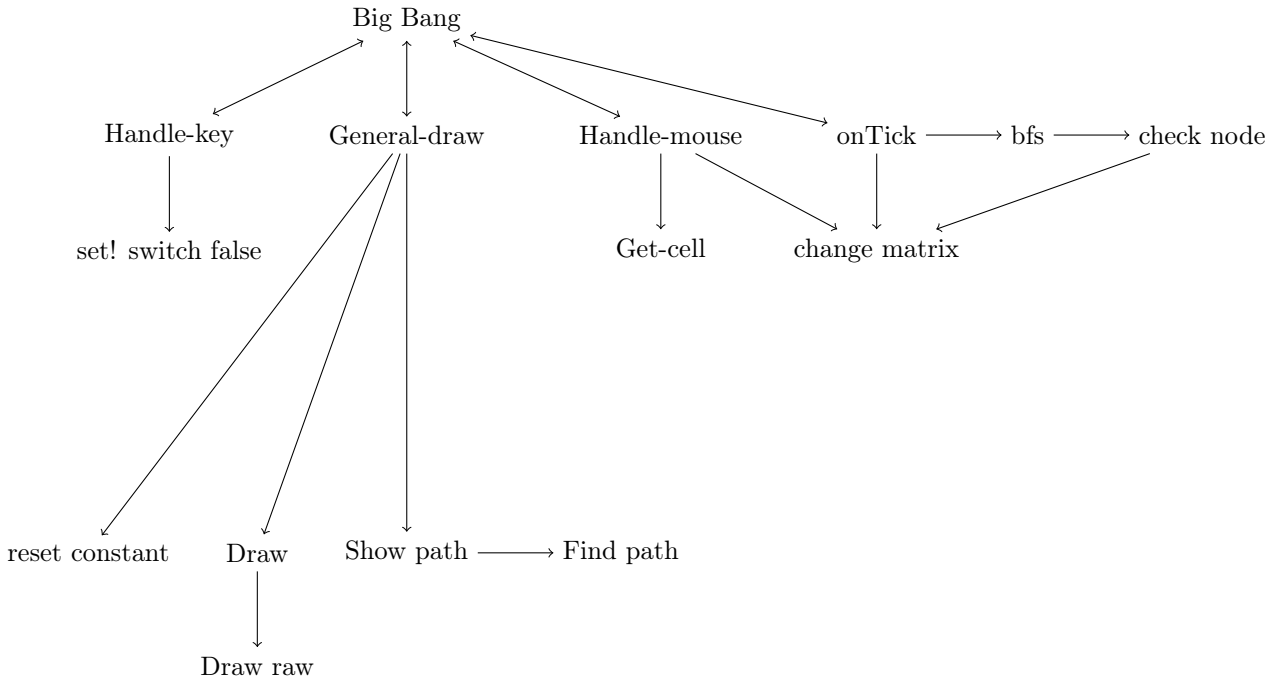
Key	Effect
"q"	quit program
"spacebar"	switch from homepage to maze
"b"	set the entry point
"e"	set the exit point
"d"	switch state to delete
"c"	clear matrix
"s"	start the solver

Table 2: Table with Actions and Effect

3.10 general-draw

(define (general-draw state) ...)
This function draw the entire maze, is particular important because handle 3 different canvas.
It draw Homepage if the constant Switch is true, otherwise combine the ‘show-path‘ and the ‘draw‘ function, and the result of this operation is the final maze

4 How functions work together



The graph above show how this function work together, note that in this scheme there aren’t all the function