Technical Documentation

Project: Navigate the Mars Rover

# **Introduction**

The aim of this project is to find the shortest path for the mars rover to navigate across all the obstacles that are present.

# **Structure of the project**

This project is organized in such way that GUI related items are available in the visual folder and algorithm in source folder and final browser distributable in library folder.

Libraries used in visualization:

* state-machine.min.js
* raphael-min.js
* jquery-1.7.2.min.js

The above said libraries are being used to create the pathfinding GUI.

**Folder Structure:**

|  |  |  |
| --- | --- | --- |
| Folder name | Sub folder | Files |
| Lib: browser distribution | nil | 1) pathfinding-browser.js  2) pathfinding-browser.min.js |
| Src: algorithms | Core  finders | 1) DiagonalMovement.js  2) grid.js  3) Node.js  4) Util.js  1)BreathFirstFinder.js |
| Visual: visualization | css  js | 1)style.css  1) controller.js  2) main.js  3) panel.js  4) view.js |

**How the application works**

The application is being developed using JavaScript on node js platform and deployed using browser distributable. The browser distributable files are built using a gulp file.

**Initialization**

* On startup (index.html file is invoked) the grid with a matrix of 36 \*24 and the floating panels is initialized.
* Three floating panels are initialized namely,

1. Instruction panel: This panel describes the instructions on how to use the application.
2. Selection panel: This panel allows us to select the option like diagonal and don’t cross corners
3. Control panel: This panel contains buttons which has functions namely start/restart, pause search, clear walls

* Initially all the cells in the grid are set to walkable which is defined as true for all blank cells. The state machine is initialized to the ready state.
* The starting position and ending position are fixed at the center of the grid.

**Selecting the starting and ending positions**

* To change the starting point, drag the starting point to the starting position which fires the mouse drag events to set the initial starting position and similarly ending position is fixed.

**Placing the obstacles**

* On mouse click (up or down) the grid will set walkable as false and hence the obstacles are created in the grid with grey color.

**Sequence of events**

* When the start-search button is clicked the state machine changes the state of that button from starting to searching which fires onsearch event.
* The onsearch event passes the parameters to findpath and the animation is shown.
* After the search is finished the state machine changes the state from searching to finished.
* The onfinish function shows the path, number of operations performed and the time taken then the drawpath function is invoked and the path is being drawn.

**Breadth first search algorithm implementation**

The algorithm has two functions namely:

1. BreadthFirstFinder: This function initializes the variables with option pass
2. Findpath: The start node and end node are initialized in this function from the grid and this function finds the shortest path.

**Initialization of the queue**

* The start point is pushed in to a queue called openlist and the starting node is marked as true.

**Execution of the algorithm**

* While the openlist queue is not empty, that node is removed and set to another variable to find the neighbors. That particular node sets it as visited.
* Get all the neighbors of that particular node through the getneigbors function which then stores the visited neighbor as parent node and sets walkable to true.
* Keeping that particular parent node as base, it checks its respective neighbors.
* If a neighbor is already visited it ignores that particular node.
* This process keeps on executing until the end node is reached.
* After the end node is reached the backtrace function is called which is present in util.js where the queue containing the neighbors is passed and it calculates the shortest path.

**Details of functions**

|  |  |
| --- | --- |
| File | Function |
| Grid.js | 1)Grid,  2) \_build nodes  3) getNodeAt  4) isWalkableAt  5) isInside  6) setWalkableAt  7) getNeighbors  8) clone |
| Node.js | Node |
| Util.js | 1) Backtrace  2) Pathlength  3) interpolate |
| BreadthFirstFinder.js | 1) BreadthFirstFinder  2) findPath |

# **References**

* qiao.github.io/pathfinding.js/visual/