MARS PATHFINDER
Engage 2020 - Mars Colonisation Program

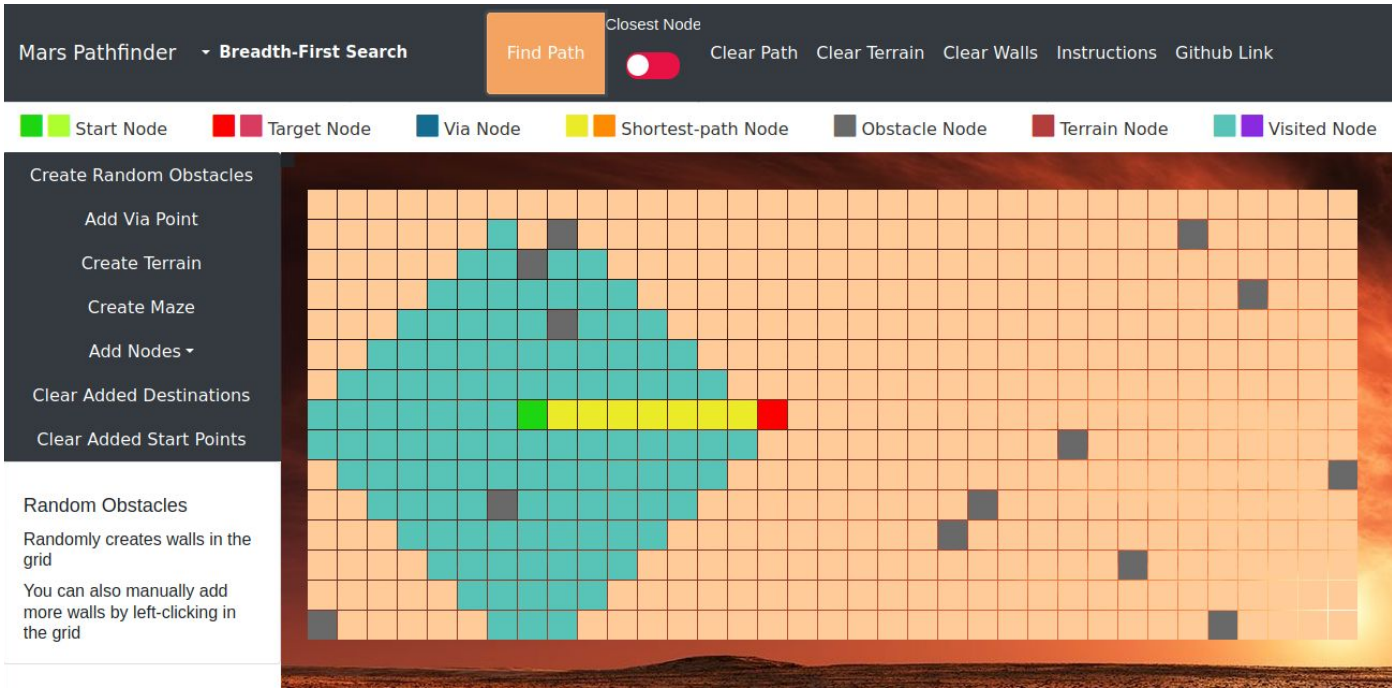
Team Martians
Gowri Lekshmy
Shivani Chepuri
Samhita Kanaparthy

Objectives:

- To create a web application to navigate a rover on the surface of Mars.
- Various pathfinding algorithms are used to find the shortest path between two points while avoiding obstacles on the way.
- Mars Exploration with the help of additional features

Link to the Application: https://marspathfinder.herokuapp.com/
GitHub link: https://github.com/gowrijsuria/MarsPathfinder

Web Application



Technology Used:

This visualization tool for path solving uses JavaScript, HTML5 with Canvas and CSS.

Browser Compatibility:

This application works on Microsoft Edge, Google Chrome, Firefox, Chromium and Safari.

Main Agenda:

Given a start and endpoint, reach the end-point in the safest and best way possible. For this, we have many algorithms and there may or may not be the best algorithm. In our case, A-Star works the best. Therefore, we are letting the user choose an algorithm.

Algorithms Used:

This provides insight into the behavior of popular traversal algorithms, such as Depth-First Search (DFS), Breadth-First Search (BFS), Greedy Best First Search, Dijkstra's Algorithm, and A* Search.

- **A* Search Algorithm**
  - A* Search algorithm is one of the best techniques used in path-finding and graph traversals. This algorithm uses heuristics like Manhattan, Euclidian, Diagonal distance to find the shortest path efficiently by approximation.
  - Time Complexity: $O(|E|)$ where E is no.of edges in the graph.

- **Breadth-First Search**
  - Breadth-first search algorithm traverses tree or graph data structures, starting at the root node and exploring all of the neighboring nodes at the present depth, prior to moving on to the nodes at the next depth level.
  - This algorithm explores paths without considering any cost function.
  - Time Complexity: $O(|V| + |E|)$ where V is no.of vertices and E is no.of edges in the graph.

- **Depth-First Search**
  - Depth-First Search algorithm traverses tree or graph data structures, starting at the root node and exploring as far as possible along each branch before backtracking.
  - Time Complexity: $O(|V| + |E|)$ where V is no.of vertices and E is no.of edges in the graph.

- **Dijkstra's Algorithm**
  - Dijkstra algorithm finds the shortest path by visiting the *closest unvisited nodes* and updates the distance of each unvisited node through it, if smaller.
  - Time Complexity: $O(|V| * \text{Log } |V| + |E|)$ where V is no.of vertices and E is no.of edges in the graph.

- **Greedy Best First Search**
  - Greedy Best First Search is an extension of BFS, using a *heuristic evaluation function* to attempt to predict how close the end of a path is to the optimal solution so that paths which are judged to be closer to a solution are extended first.
  - This is also called the *Pure Heuristic Search* algorithm.
  - Time Complexity: $O(|V| * \text{Log } |V|)$ where V is no.of vertices in the graph.

Additional Features:

- Some features are added to simulate some situations and use cases that a rover can encounter on Mars Surface. These features are also user-friendly making navigation through the app easy.
- INSTRUCTIONS tab on the Top Navigation Bar, on click, displays the basic introductory instructions.
- HOVER OVER BUTTONS also helps understand what each button does and how to use it.
- For new features, we are also displaying some information and ideas behind those features in the BOTTOM LEFT.

Some additional ways that we implemented to explore the app and more of Mars robustly:

- **Random Obstacles** - This feature generates random obstacles on the surface of Mars. These obstacles are not manoeuvrable. This feature helps the user avoid creating obstacles on his/her own while using the app.

- **Create Maze** - This feature uses a Recursive Maze division to generate a maze. The idea behind this feature is to simulate a typical Mars Terrain where most of the paths are unmanoeuvrable.

- **Via Point** - A Via Point is a Drop-by point before reaching the destination. It is similar to having multiple destinations with a fixed Final Destination. Traverse through a via point before reaching your destination.

- **Create Terrain** - Terrain represents craters, mountains, etc landforms on Mars which are dangerous but manoeuvrable. It assigns different costs to different nodes randomly to simulate the environment. It basically creates a weighted graph.

- **Add Multiple Destinations** - Creates multiple destination points for the rover. The rover in most of the real scenarios will have to keep travelling all the time from one destination to the other. Given, where the rover set out on its journey (the start node) and what all places it needs to reach, we can predict the best possible path to reach all the destinations using TSP.

- **Add Multiple Start points** - Creates multiple start points for the rover simulating a multi-agent system. In a real scenario, we may have more than one rover, in which case, not all tasks need to be taken up by one of the rovers, nor all rovers to take care of one task. Distribution of tasks among them is critical. This idea and feature is the first step towards a more complicated multi-agent system.

- **Closest Node -** Finds the closest destination when multiple destinations are present. Chooses one start point among many to reach the destination as quick as possible. This is choosing the rover that is closest to the destination among all the rovers. This feature is important and specific to a case where all the working rovers receive an instruction to reach a destination as soon as possible.

- **Travelling Salesman Problem with multiple destinations -** Finds the optimal path to follow when there are multiple destinations.

Instructions:

- Change the start and destination node positions by dragging
- To draw walls, use left-click
- To delete walls, use right-click
- Click on create Terrain to create a Weighted Graph

- **<u>Normal Pathfinding</u>**
    - Has one start node and one end node
    - Toggle **Closest Node** to OFF
    - Select the algorithm by using the dropdown list
    - Click on **Find Path** for visualising the path followed by the rover

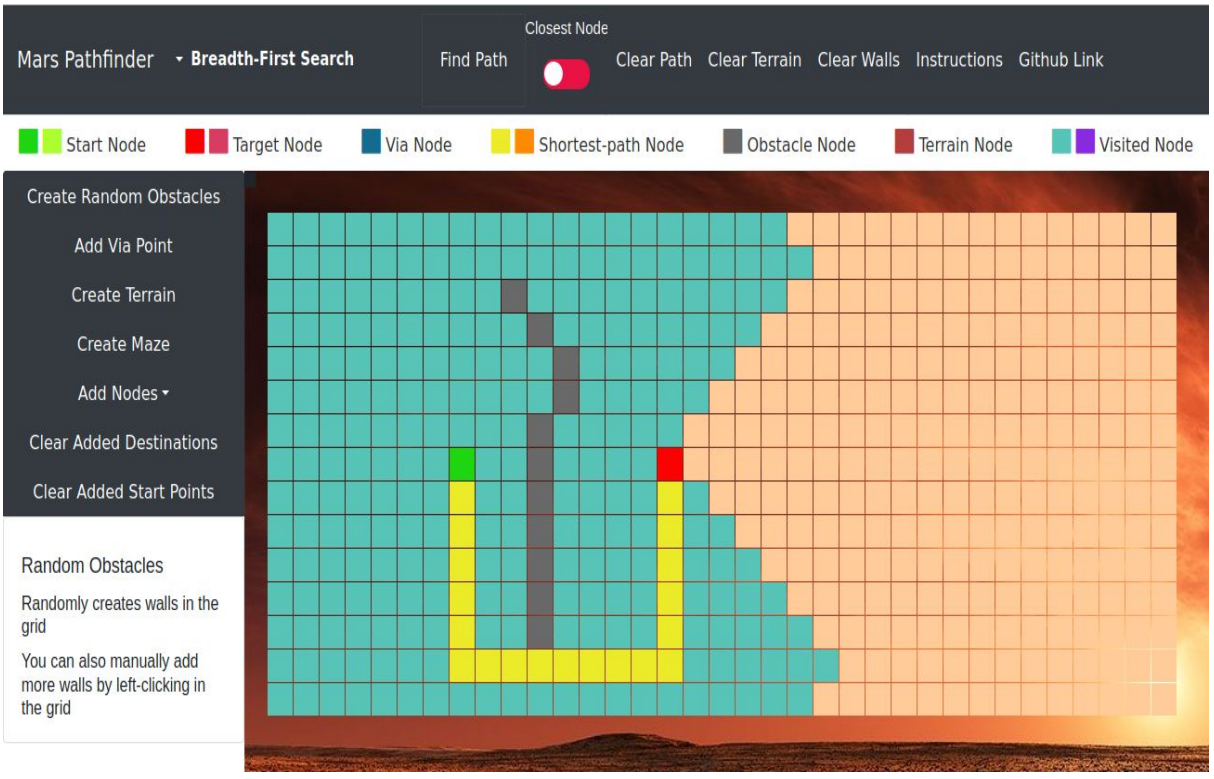
- **<u>Complex Pathfinding</u>**

    - Add multiple destinations or start nodes with via points for complex pathfinding
    - Click on **Add Via point**, drag it to a suitable point and click **Find Path.** We have limited Via Point to only one. To clear the Via Node, Click on the button **Clear Via Node.**
    - Click on **Add Nodes**, then click on the surface to create ***an extra start or destination node***. The user can click to add as many start or destination nodes as possible.
    - Toggle the **Closest Node** button to GREEN for finding the closest path and to RED for TSP with multiple destinations.
        - **(Toggle ON/Green)** Travel to your **closest destination** on Mars either with ***multiple starts or destination nodes***
        - With multiple destination nodes, finds the closest destination
        - With multiple start nodes, chooses the closest start point to the destination
        - **(Toggle OFF/Red)** Travel through all destinations optimally with the help of ***Travelling Salesman Problem***

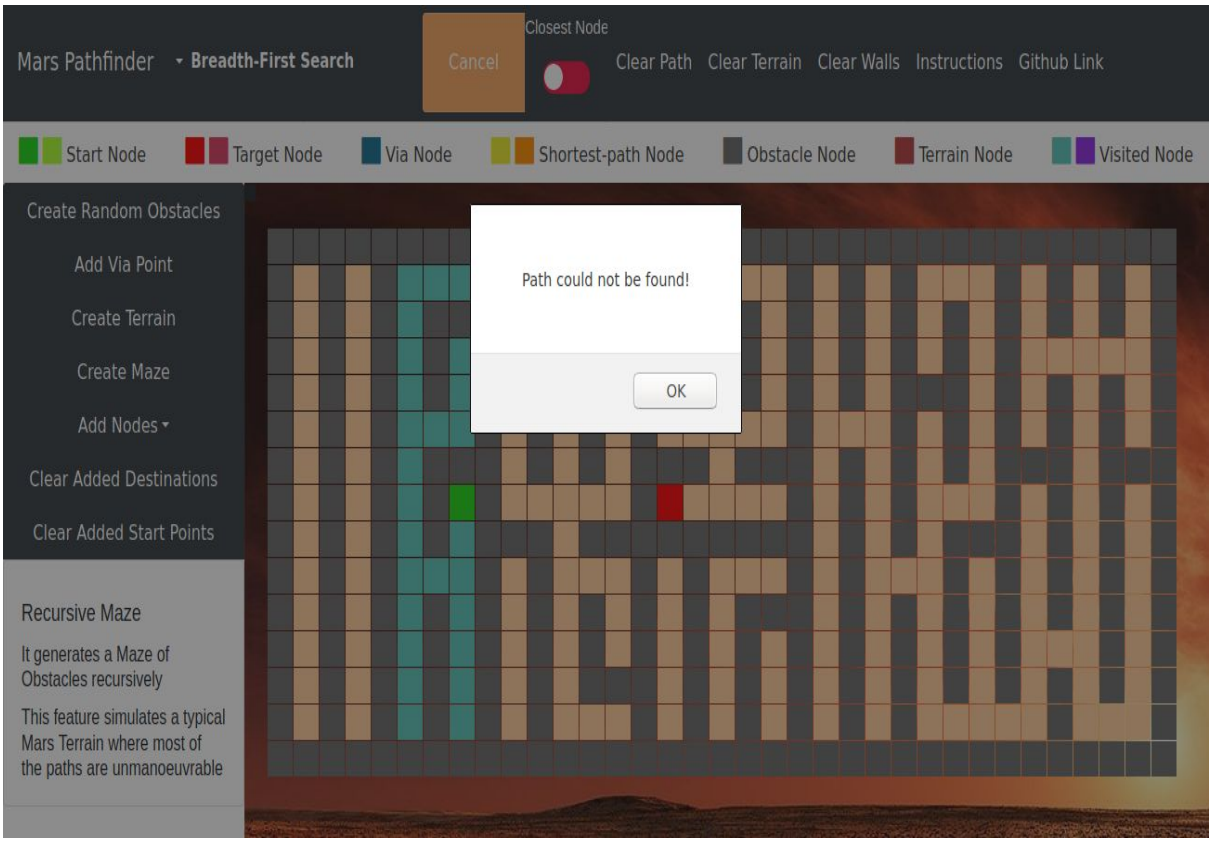
    - Click **Create Terrain** for simulating the Martian surface
        - In the presence of TERRAIN nodes, the rover follows an optimal path to reach the destination minimising the cost.
        - Please note that all functionalities and features can also be implemented with TERRAIN nodes in the grid.
    - Select the algorithm using the dropdown list
    - Finally, click on **Find Path** for visualising the path followed by the rover

Working Functionalities:

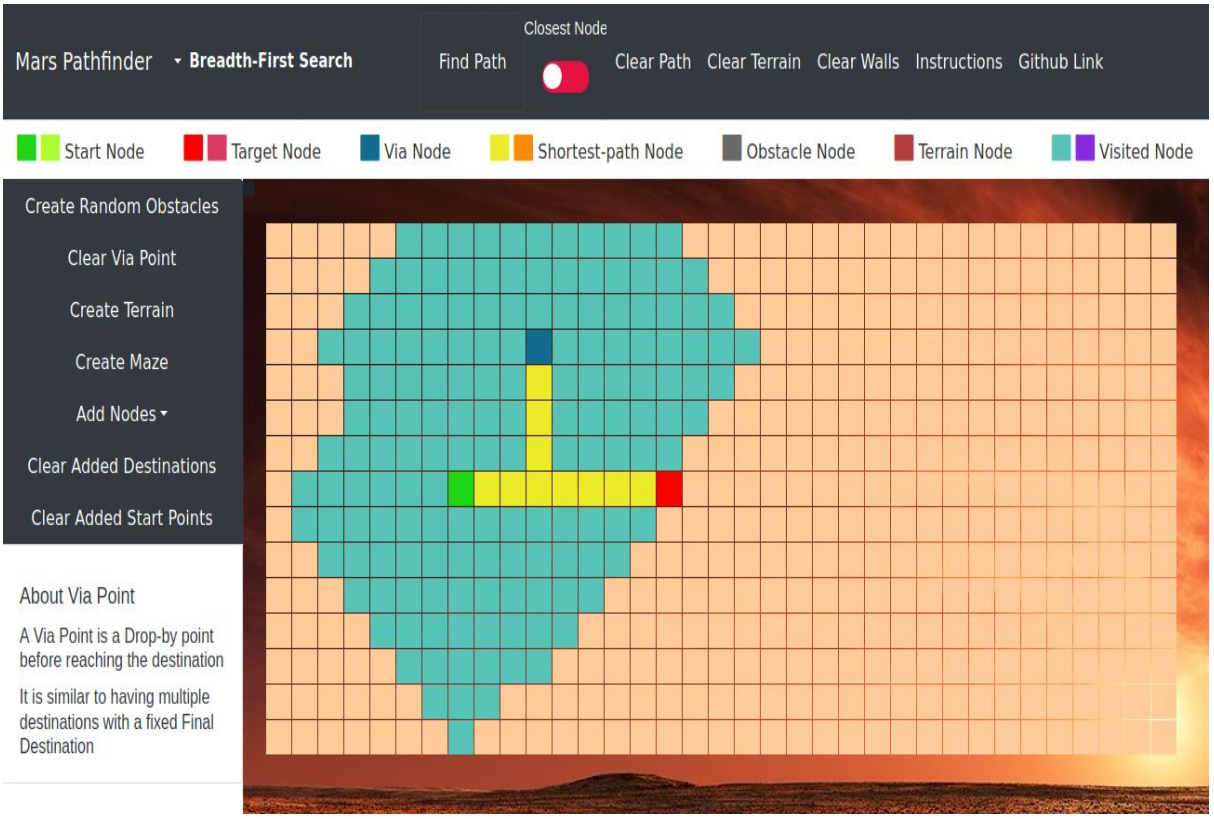
- **Basic Path Finding:**



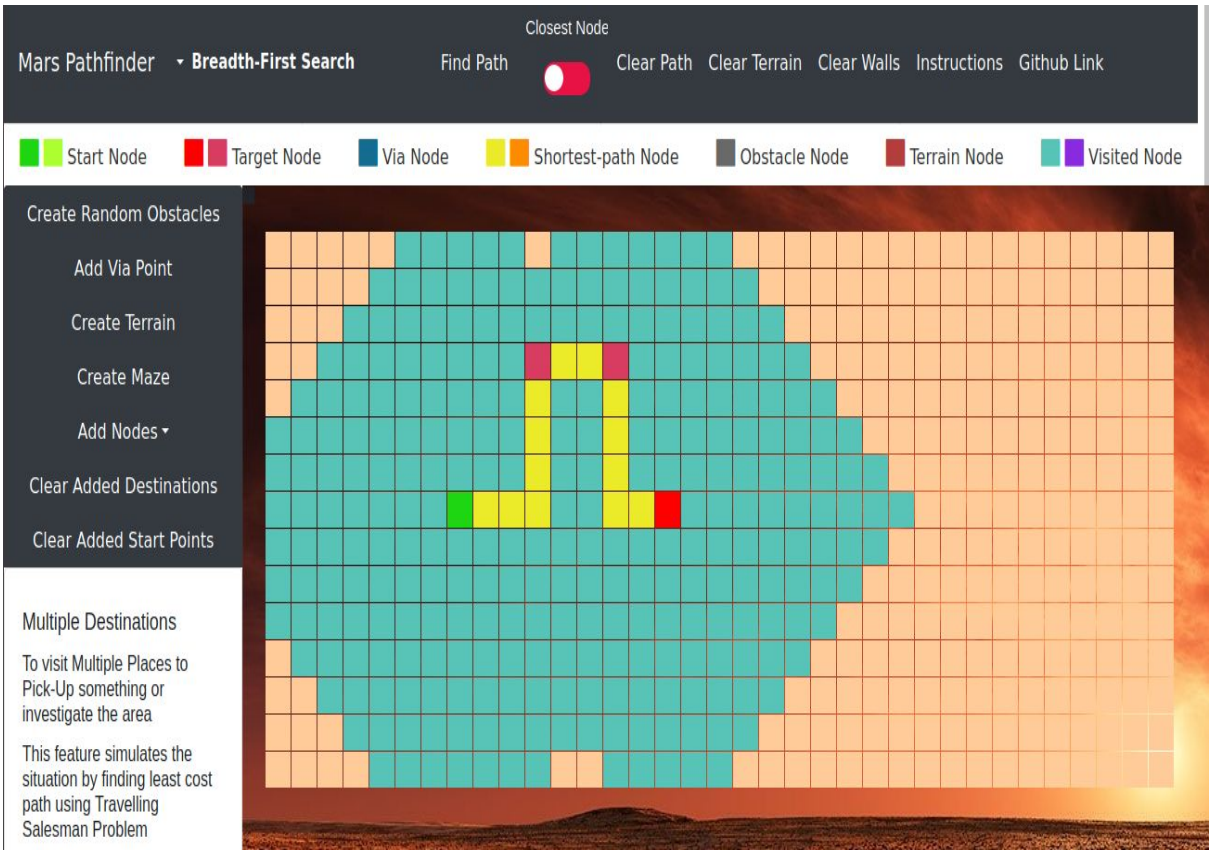
- **Random Obstacles / Maze :**



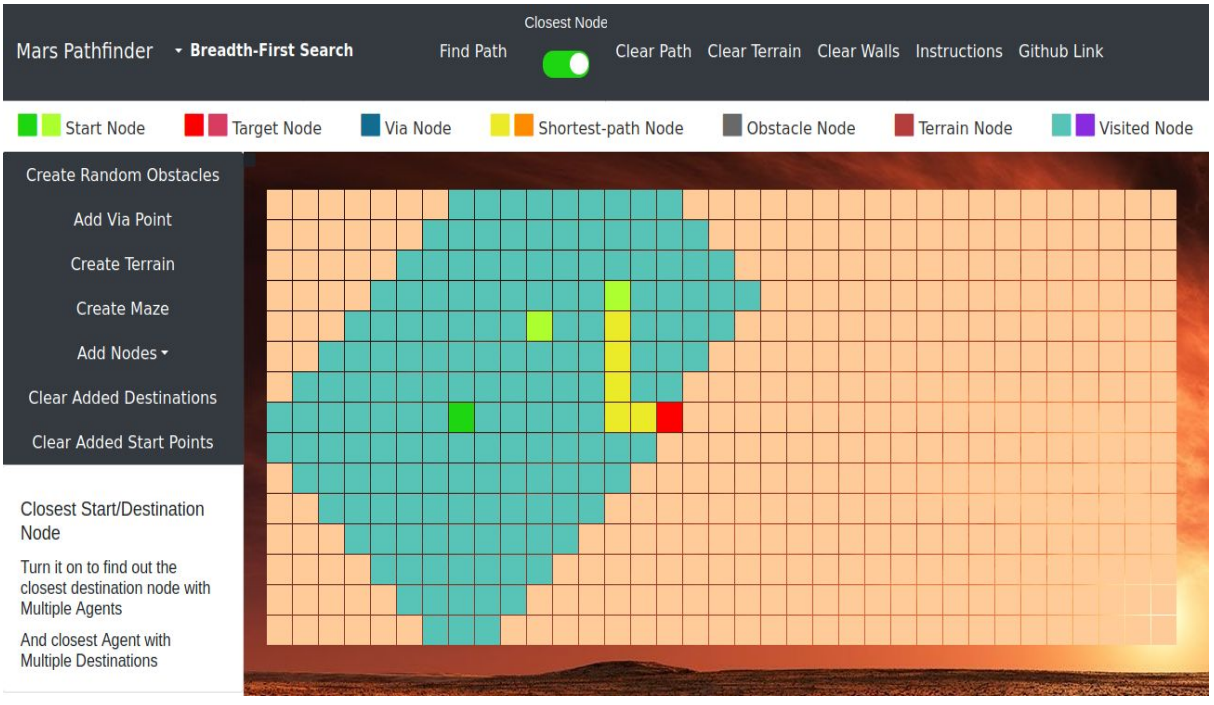
- **Via Points:**



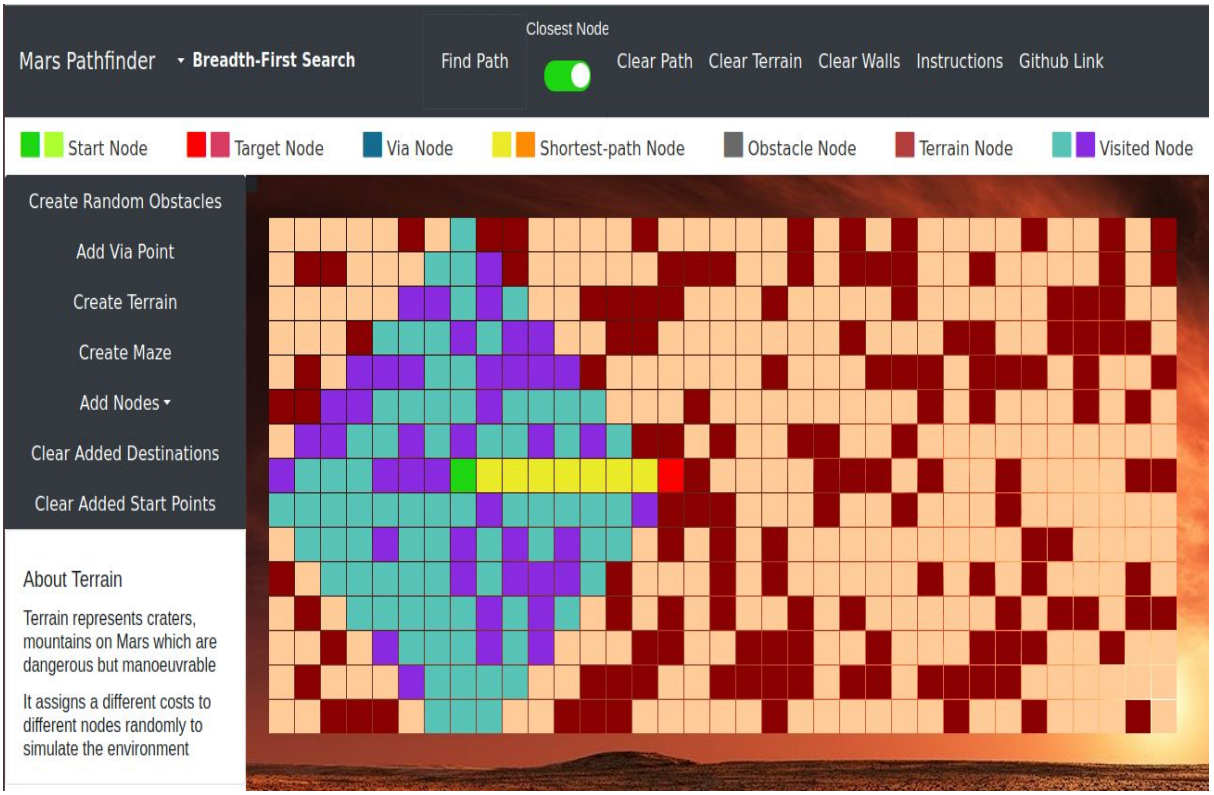
- **Multiple Destinations (TSP):**



- **Multiple Start Nodes:**



- **Terrain:**



Hosting Platform:

This application is hosted using **Heroku** with Continuous Integration from GitHub.

Ideas and Future Work:

- We are planning to extend this work by creating moving obstacles which have to be avoided during pathfinding.
- We also intend on making the start node draggable while the search is going on. This feature can get close to the real scenario where the Agent is already on a mission to reach a destination and it encounters a new instruction somehow to divert its path or to go to a new destination.
- Bootstrap and CSS may be used more to make the page more responsive to screen-size and window-size.
- Improve on better strategies, features and functionalities in a Multi-Agent Scenario.