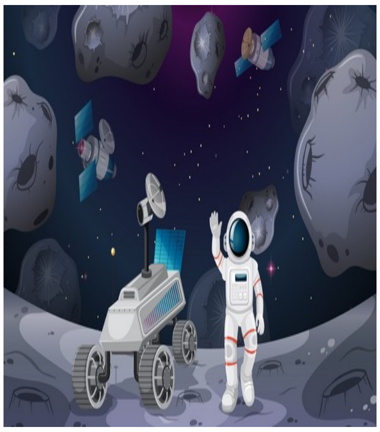
Technical Documentation

Release 0.1

Made by: Sriram Nuthi
Source code hosted on github: https://github.com/Nuthi-Sriram/NativeMarsRover
Website URL: https://nativemarsrover.web.app/
Email: sriramnuthi@gmail.com
github: https://github.com/Nuthi-Sriram

Video Demo of the project : https://youtu.be/FD2ArG5F7I0 **(Please Watch!!)**

Project- Native Mars Rover



July 22, 2020

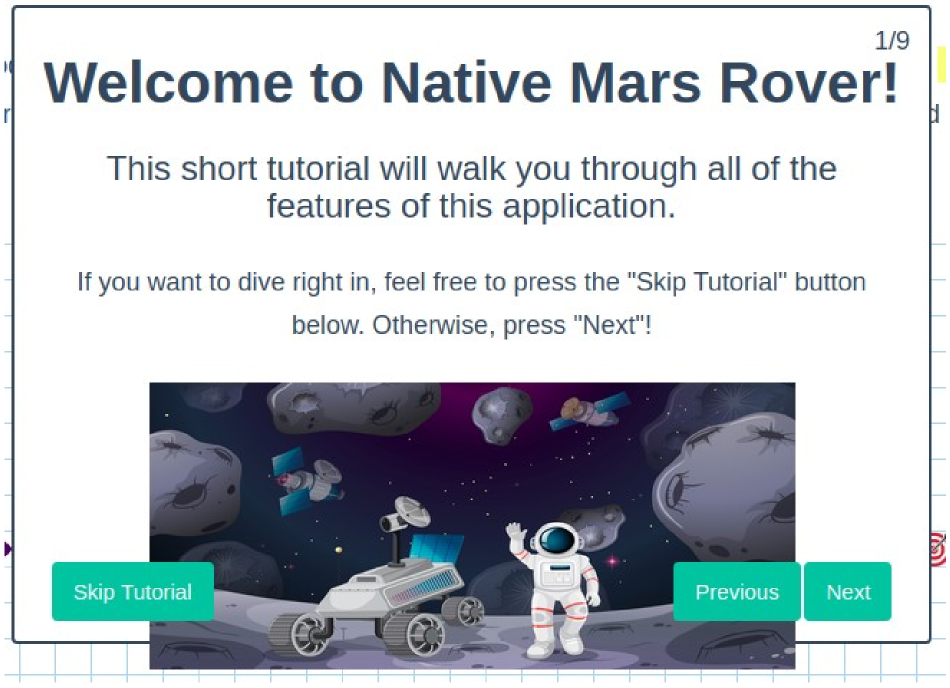
Index

Okay, let's get to web development and the project I made and the experience I got making it.

The project is called Native Mars Rover, aptly because it has been built keeping in mind the real life scenarios that an Actual mars rover could face when it is exploring mars, taking the practical scenario into account the keywords used in the UI have been designed.

Note: I was inspired to do this project after the demonstration of the https://qiao.github.io/PathFinding.js/visual/ project by Vivek Sir. And this project is also inspired by the website built by Clement Mihailescu. I have added additional features, and heuristics approaches to the algorithm, So the user can pick a particular heuristic function as per his/her needs.

When the project is rendered on the screen first you get a modal which is a tutorial that walks you through how to use the application.


1/9

Welcome to Native Mars Rover!

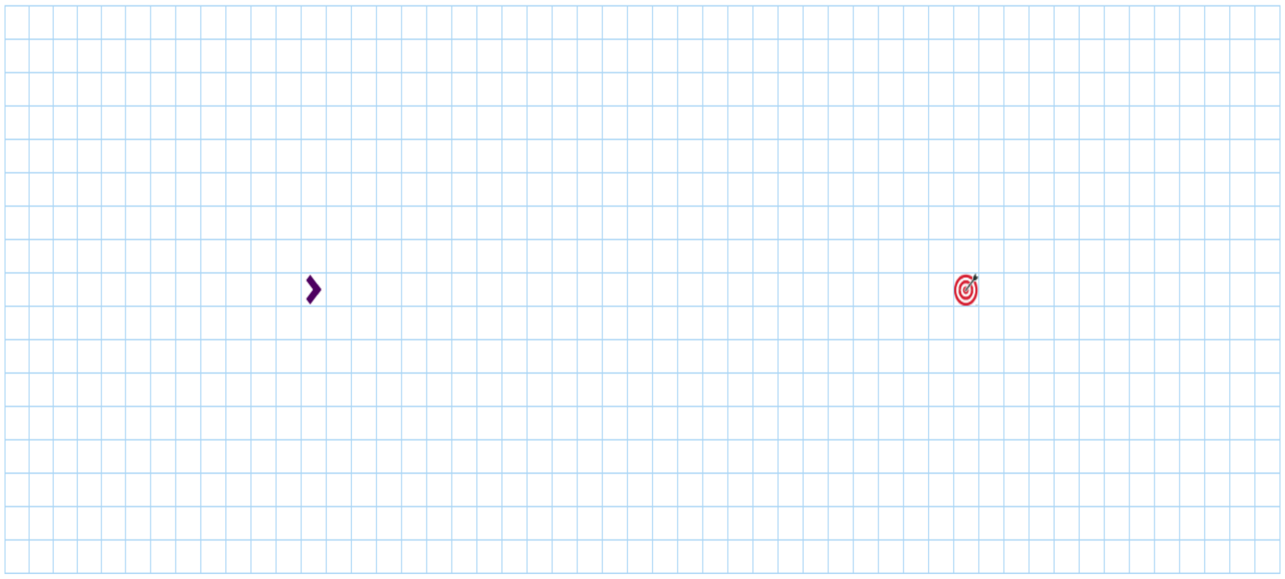
This short tutorial will walk you through all of the features of this application.

If you want to dive right in, feel free to press the "Skip Tutorial" button below. Otherwise, press "Next"!

The tutorial could also be activated in between the usage of the application if the user needs it by clicking on the project title

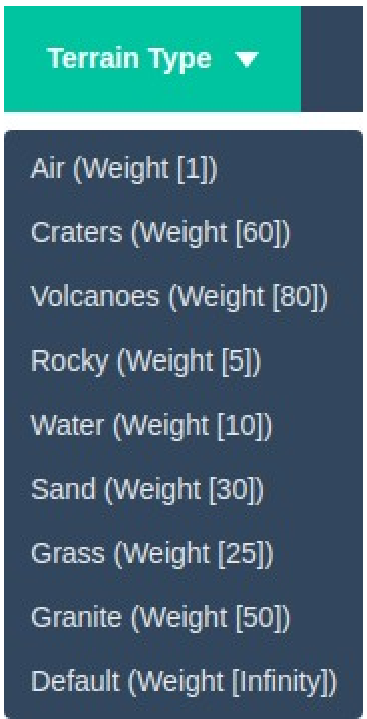
Which re-renders the tutorial if skipped earlier.

This is the first html element you will see on the page. Okay, not the first box, but anyways. Press the drop-down and you can select an algorithm from the available ones. Right now I have built ten algorithms and I am planning to add more in the coming days.



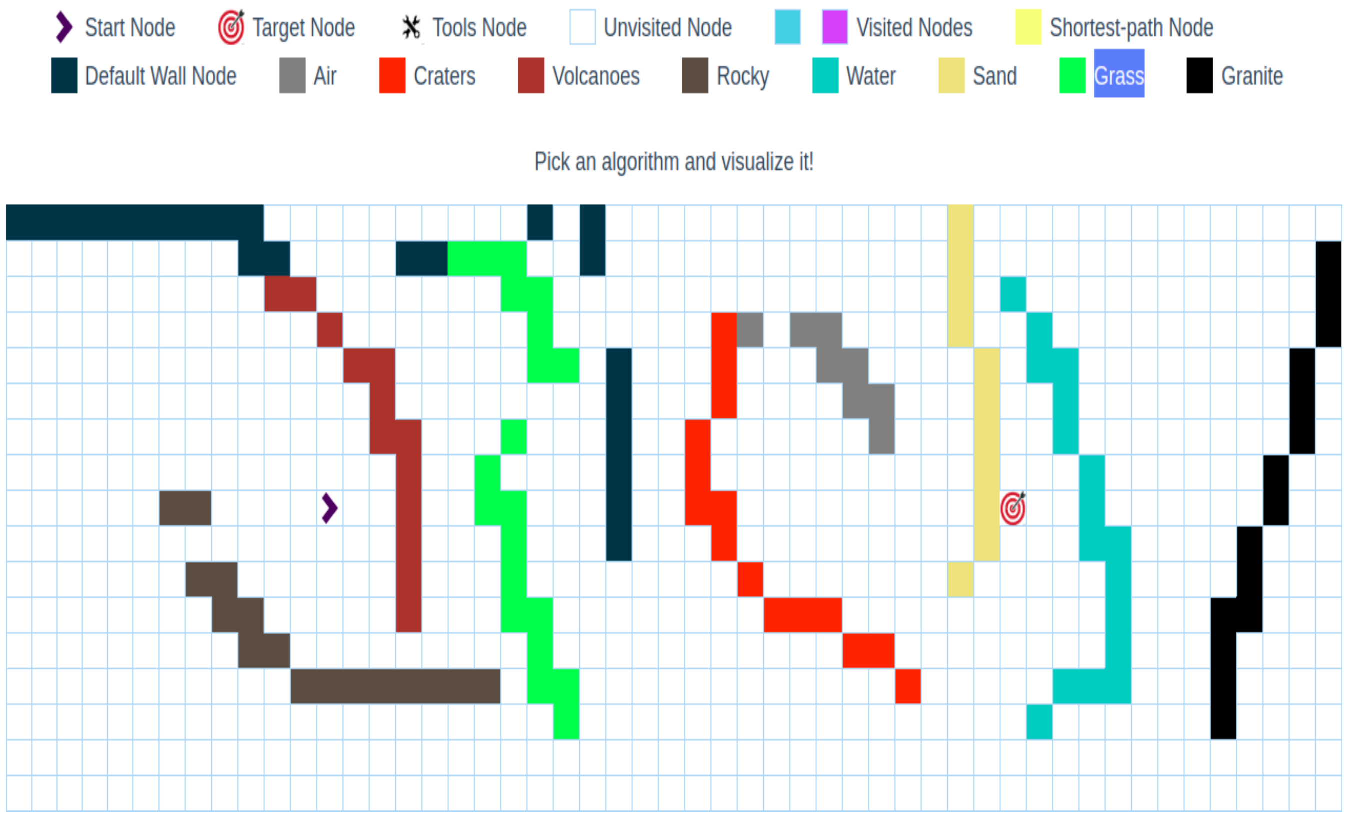
This is the main canvas where everything will happen. The Purple arrow represents the source and the red color Arrow-Dart represents the destination. Now I can travel in all four directions to reach the destination.

To make it more interesting, you can add obstacles in between which needs to be avoided during the traversal.



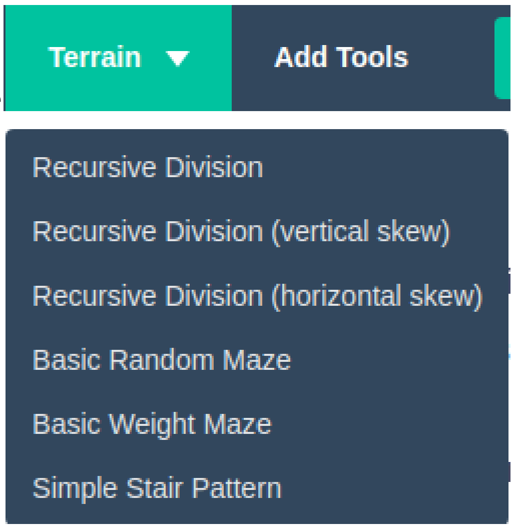
Here the user gets to choose the type of terrain or obstacles he/she can add, the default terrain is a wall which is impermeable the weight on the wall can be taught of being as infinity(The rover can’t get through the wall). The rest of the terrains the user gets to chose have a weight assigned to them. That means the rover could pass through the terrains and obstacles but it would incur as cost of having to pass throught them. The user can pick any terrain of his or her choice.

A Glimpse of how the canvas would look with different obstacles.



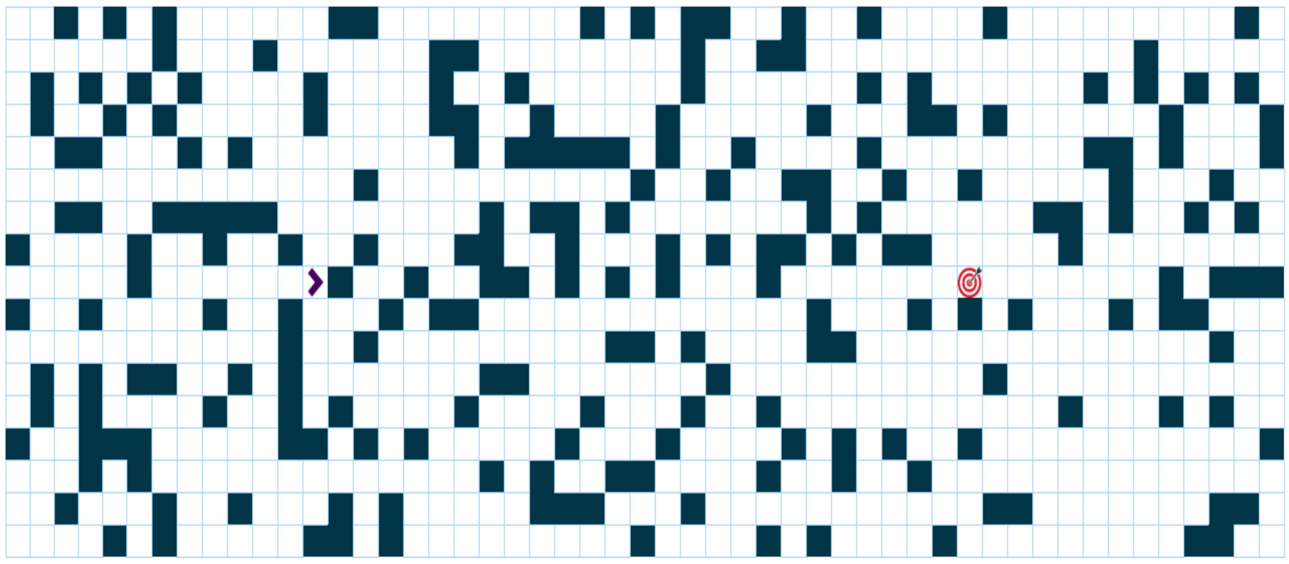
Coming to the next feature:

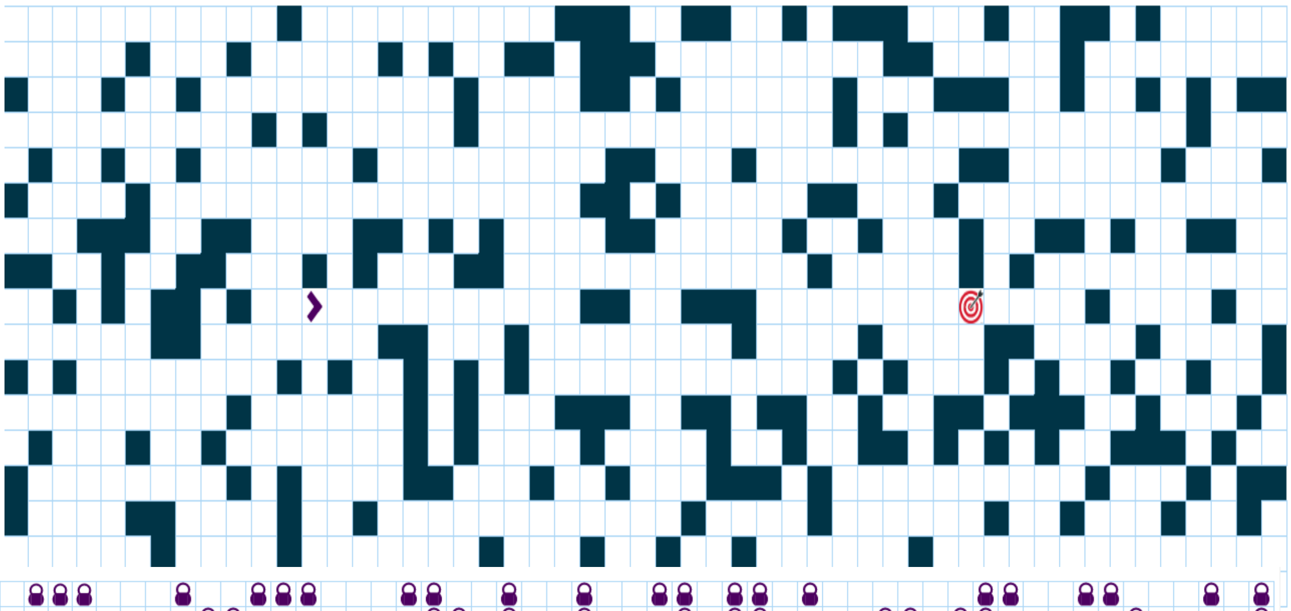
If the user doesn't want to make any such obstacles there are ready made obstacles under the **Terrain** drop-down menu. The 4th option (Basic Random Maze) is the option which generates random

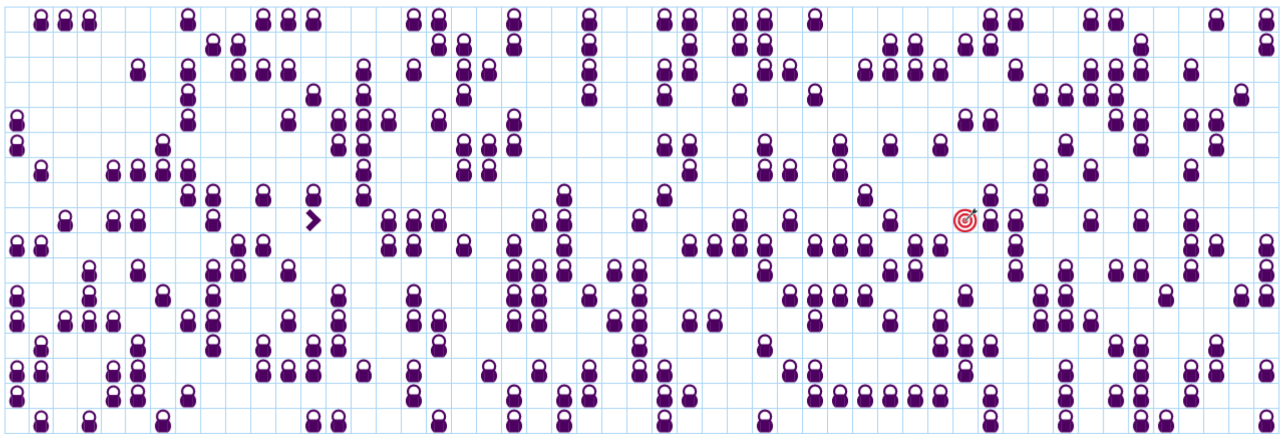


terrain each time a user chooses it. The rest are predetermined terrains or obstacles which are ready-made and the user gets to choose which one to pick in order to do the test.
For Example: Recursive Division(Vertical Skew) is where vertically obstacles are set up and certain algorithms go well, like in this case depth first search algorithm would be a good choice.
Recursive Division(Horizontal Skew) is where horizontally obstacles are set and algorithms like Breath first search algorithms do well here in Average case scenarios.

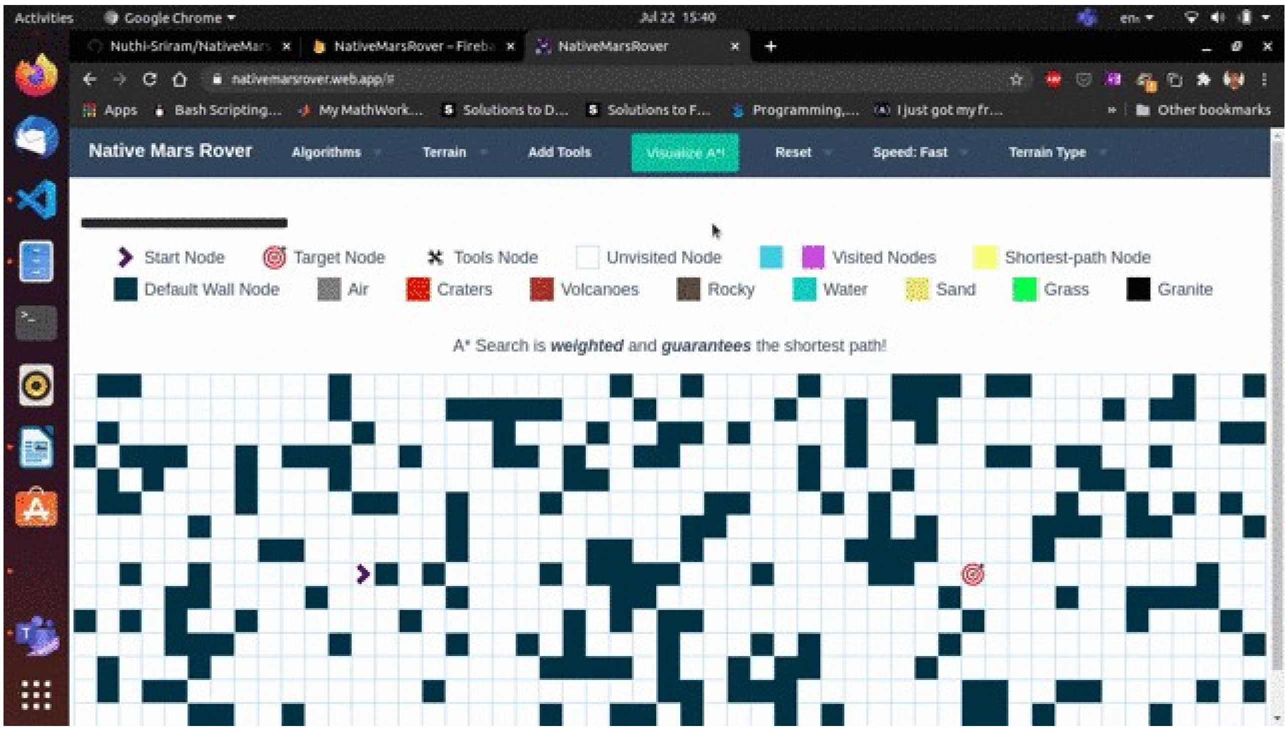
On Clicking Basic Random Maze







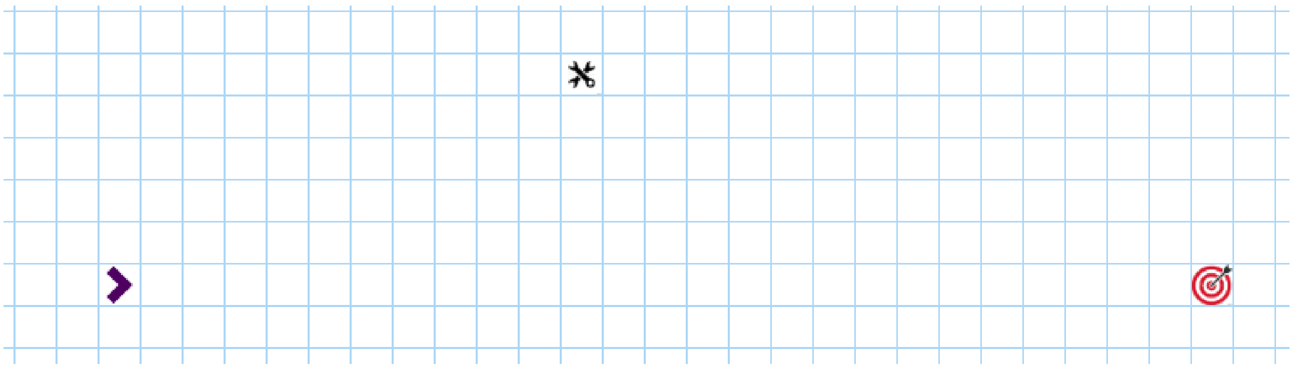
Simple demonstration of A* Algorithm in execution!!(GIF)

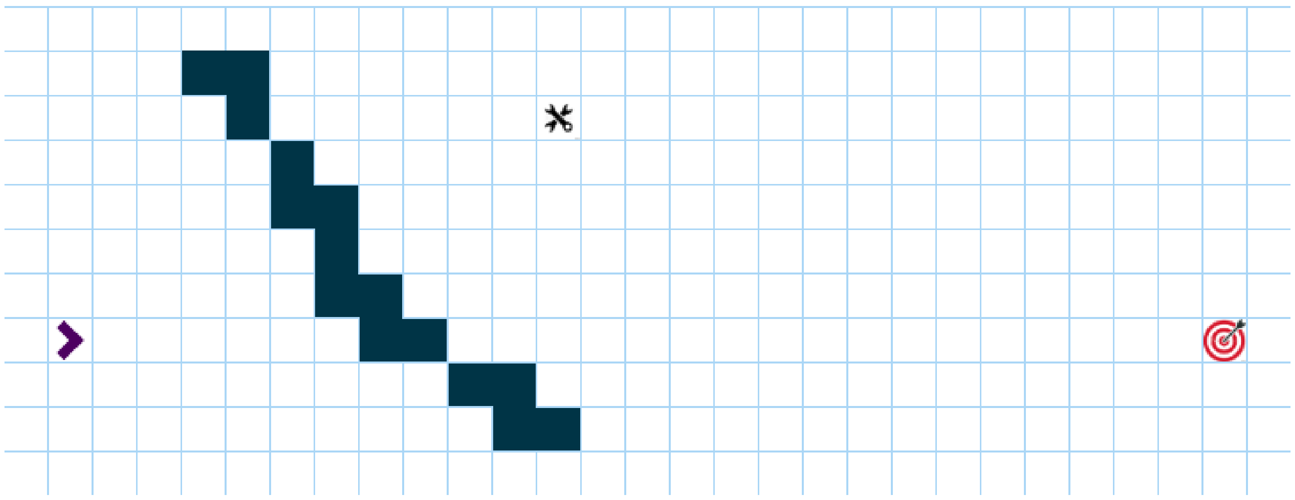


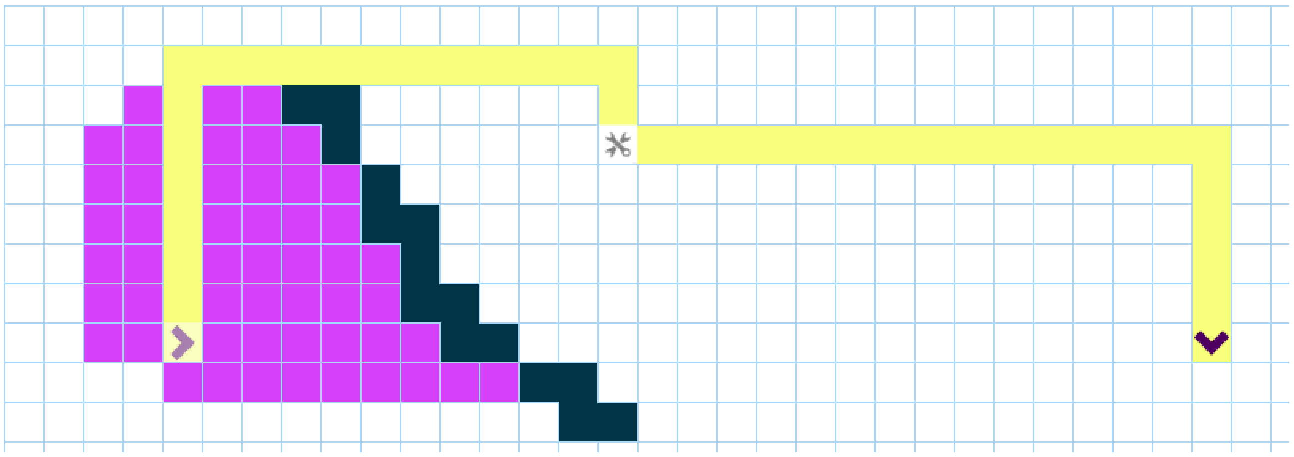
A* Search is *weighted* and *guarantees* the shortest path!

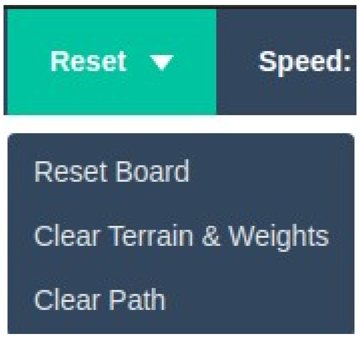
We can just make the algorithm perform more, by adding another node that is the tools node, The algorithm can handle two destination nodes i.e., it can visit the first destination node and then visit the final destination node.

This option adds another node which the algorithm has to visit before reaching the destination node.(The rover needs to pickup the tools needed and deliver the tools to the destination camp).



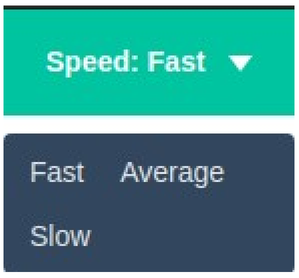




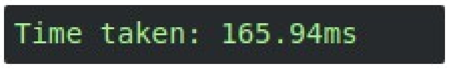


Want to start over once again!! No problem there are 3 options under reset where option 1: resets the entire board and nodes to there default positions, option 2: Clears the obstacles or the terrain and the weights if any, and option 3: Just plainly clears the path that the algorithm has traversed.

You also have an option to adjust the speeds at which the visualization animations are rendering on the screen!!



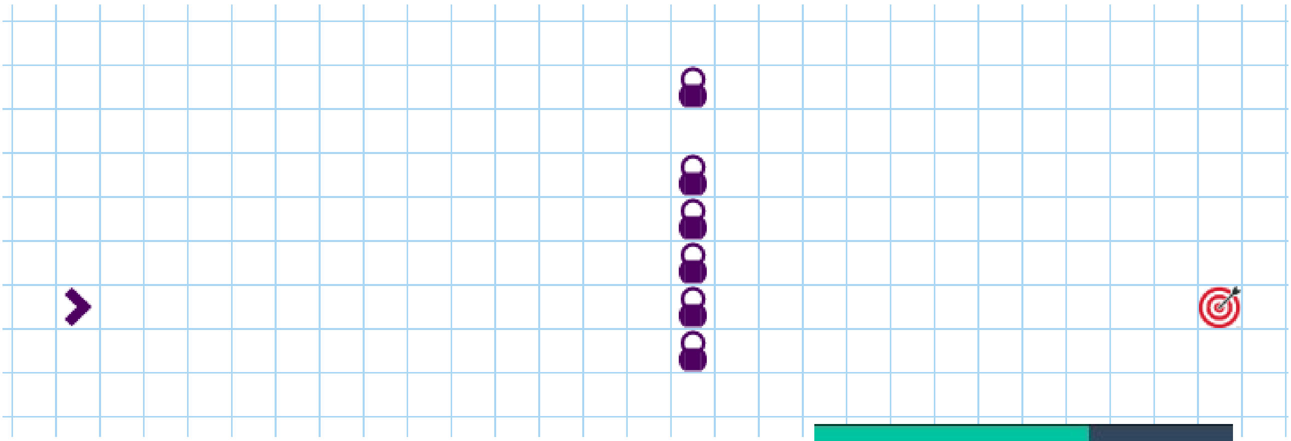
Once the algorithm has finished executing the application renders how long the algorithm has taken to reach the destination node in milli seconds. Which is useful when the user wants the check how long a particular algorithm took to reach the destination node, and compare the algorithms based on the time interval thus, helping the user decide which algorithm is the best for a given scenario.



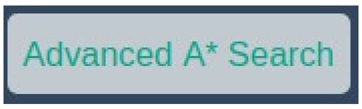
```
Time taken: 165.94ms
```

You could add weight for weighted search algorithm!!

This can be done by pressing down the “W” key and simutaneously drawing on the canvas.

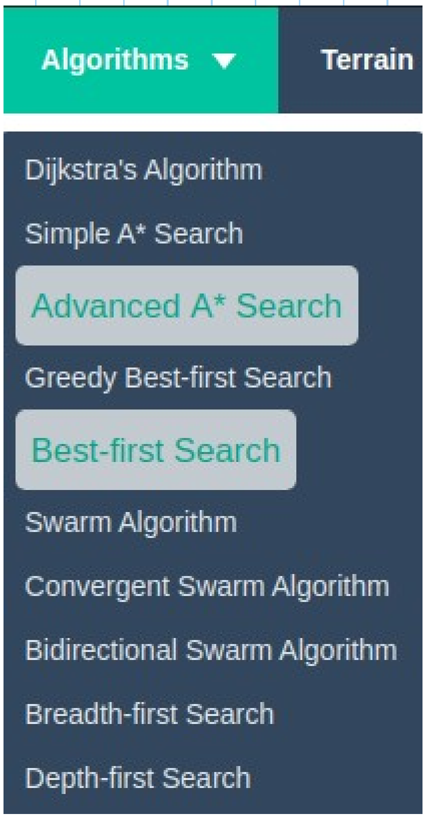


You can notice two types of CSS over here in the algorithms
The plane navy blue css is for direct algorithms which are direct click and run.
For advanced algorithms there is a different CSS



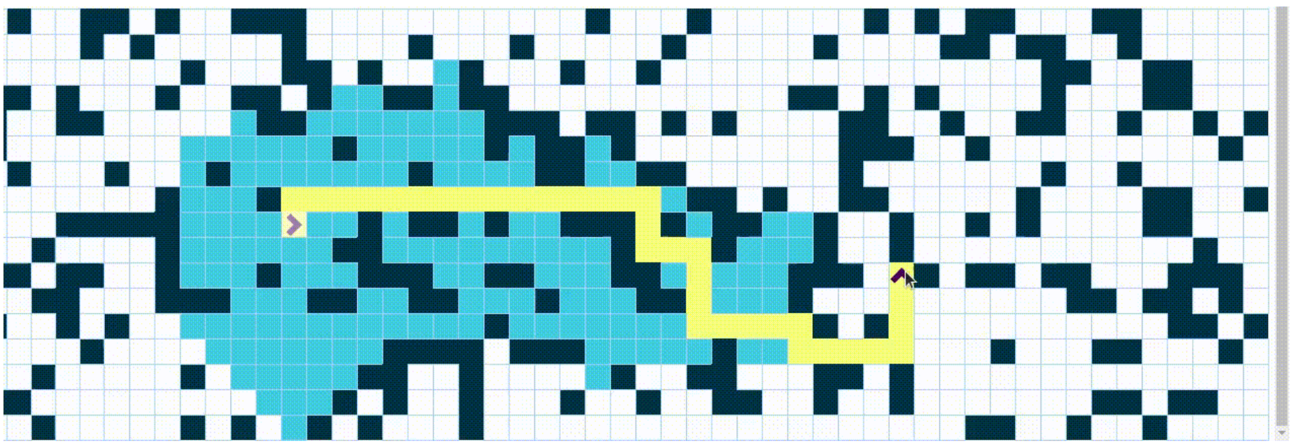
Which denotes that when clicked on this algorithm the user gets to choose which Heuristic function they would like to run. A popup modal appears which allows are user to choose their function of interest and then click on visualize.

Note: All simple versions of the algorithms are by default making use of manhattan function.



Dynamic UI Behaviour

You can dynamically keep shifting the destination once the algorithm has been executed which gives a real time detail on what path the algorithm would have taken otherwise.



Technology Stack used :

Front End:
HTML, CSS, Bootstrap, javascript

Libraries:
Node.js, jquery

FrameWorks:
Firebase cloud functions

Version Control:
Git

Source code:
Github

Hosting:
Firebase hosting

For the front end of the application html5, css3 has been used, for execution of the animations and document queries have been done using javascript. All dom manipulation realted operations are done using javascript. Boostrap library has been used for the UI appearing on the screen.

Although the project doesn't need a backend, the setup and running of the server is using node.js library.

Testing the application and running them is through firebase cloud function.

Finally the project is hosted on Firebase.

Installation and running of the application

Clone the repository:

```
git clone https://github.com/Nuthi-Sriram/NativeMarsRover.git
```

Project Setup:

```
npm install
```

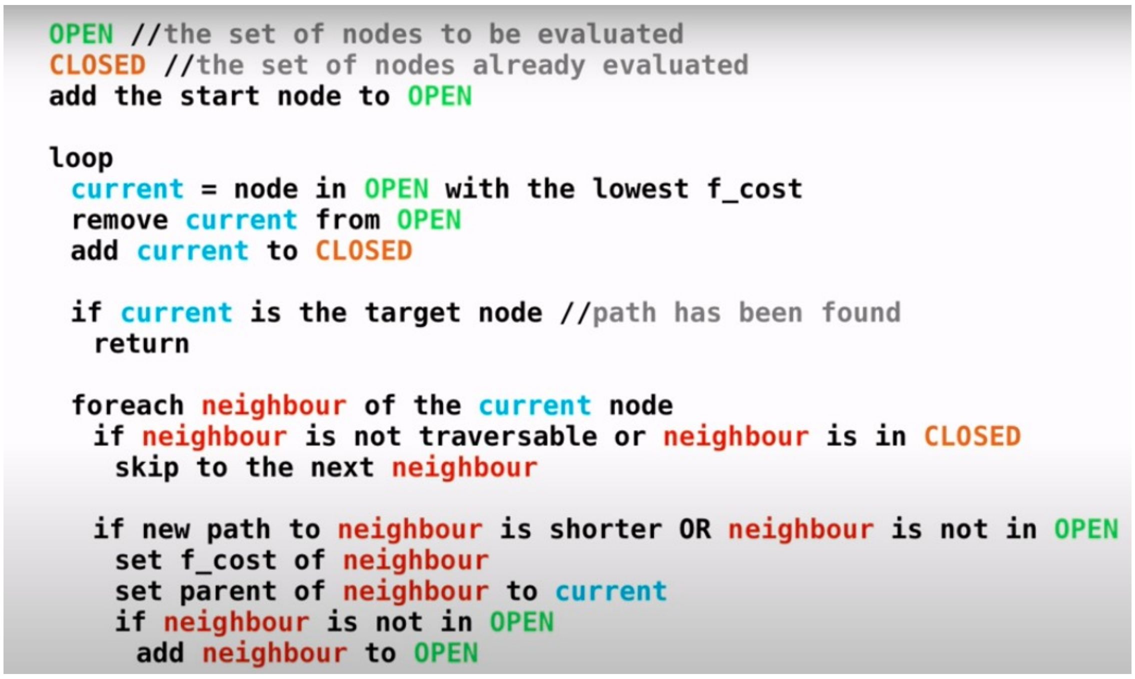
Compiles and hot-reloads for development:

```
Firebase serve
```

Note: In cace of unresponsive behaviour from the app, hard-reload (Ctrl+F5).

Approach of arriving at the AI implementation of A* PathfindingAlgorithm

Step 1: Pseudo code



```
OPEN //the set of nodes to be evaluated
CLOSED //the set of nodes already evaluated
add the start node to OPEN

loop
  current = node in OPEN with the lowest f_cost
  remove current from OPEN
  add current to CLOSED

  if current is the target node //path has been found
    return

  foreach neighbour of the current node
    if neighbour is not traversable or neighbour is in CLOSED
      skip to the next neighbour

    if new path to neighbour is shorter OR neighbour is not in OPEN
      set f_cost of neighbour
      set parent of neighbour to current
      if neighbour is not in OPEN
        add neighbour to OPEN
```

Resources from where I have learnt the concept required for the project:

http://theory.stanford.edu/~amitp/GameProgramming/Heuristics.html

https://raw.githubusercontent.com/SebLague/Pathfinding/master/Episode%2001%20-%20pseudocode/Pseudocode

http://diposit.ub.edu/dspace/bitstream/2445/140466/1/memoria.pdf

https://www.sciencedirect.com/science/article/pii/S1000936116301182#:~:text=The%20octile%20distance%20is%20used,distance%20between%20two%20cells%20heuristically.&text=The%20octile%20distance%20between%20p,%7C%20y%20%2D%20y%20%E2%80%B2%20%7C%20.

http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html

https://www.youtube.com/watch?v=tvAh0JZF2YE

https://www.youtube.com/watch?v=-L-WgKMFuhE&t=36s

Projects which I drew inspiration from:

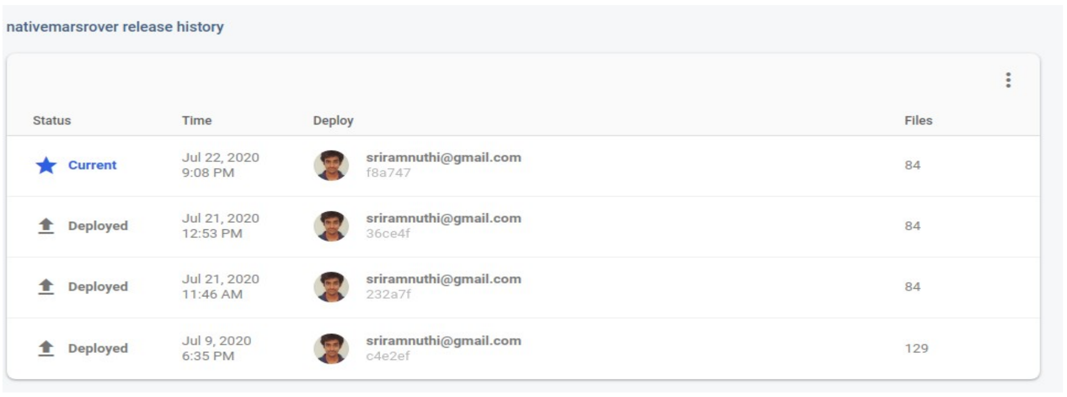
https://codepen.io/stevepepple/pen/myqwRY

https://qiao.github.io/PathFinding.js/visual/

https://tinku10.github.io/pathfinding-visualizer/

Contributors to the project can be viewed here:

https://github.com/Nuthi-Sriram/NativeMarsRover/graphs/contributors



nativemarsrover release history

Status	Time	Deploy	Files
Current	Jul 22, 2020 9:08 PM	sriramnuthi@gmail.com f8a747	84
Deployed	Jul 21, 2020 12:53 PM	sriramnuthi@gmail.com 36ce4f	84
Deployed	Jul 21, 2020 11:46 AM	sriramnuthi@gmail.com 232a7f	84
Deployed	Jul 9, 2020 6:35 PM	sriramnuthi@gmail.com c4e2ef	129

TODO:

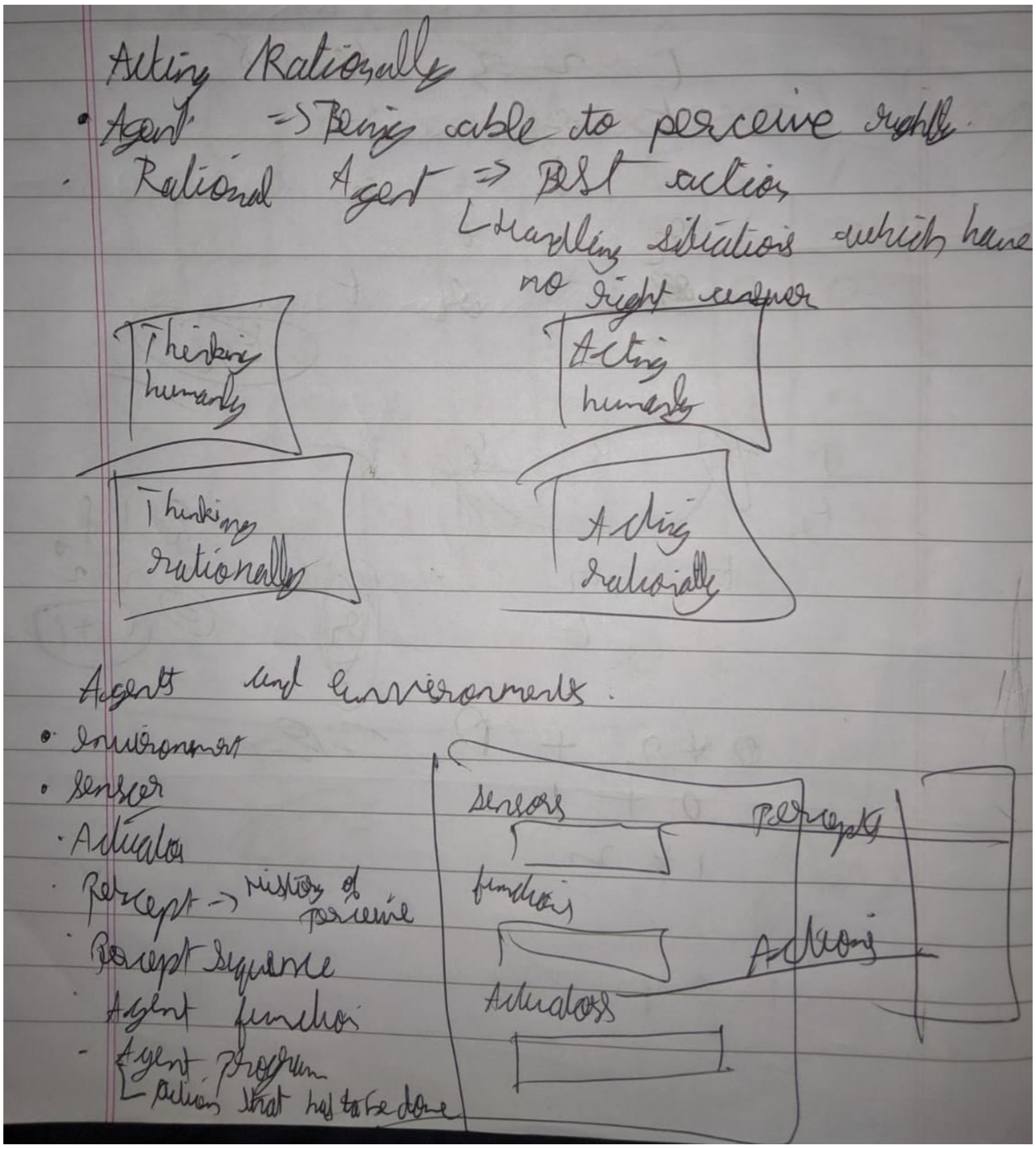
1) Adding 2 more features, number of operations and path length to get to the destination.

2)Removing few bugs which still persist in the app which prevent more then one type of terrain from being on the canvas.

3)Optimising the code as per lighthouse.

4)Github actions to deploy the app automatically.

Invaluable learnings from the Webinars which I have applied to the project:



Acting Rationally

- Agent => Being able to perceive rightly.
- Rational Agent => Best action
  - Handling situations which have no right answer

Thinking humanly

Acting humanly

Thinking rationally

Acting rationally

Agents and Environments:

- Environment
- Sensor
- Actuator
- Percept -> history of perceive
- Percept Sequence
- Agent function
- Agent Program
  - action that has to be done

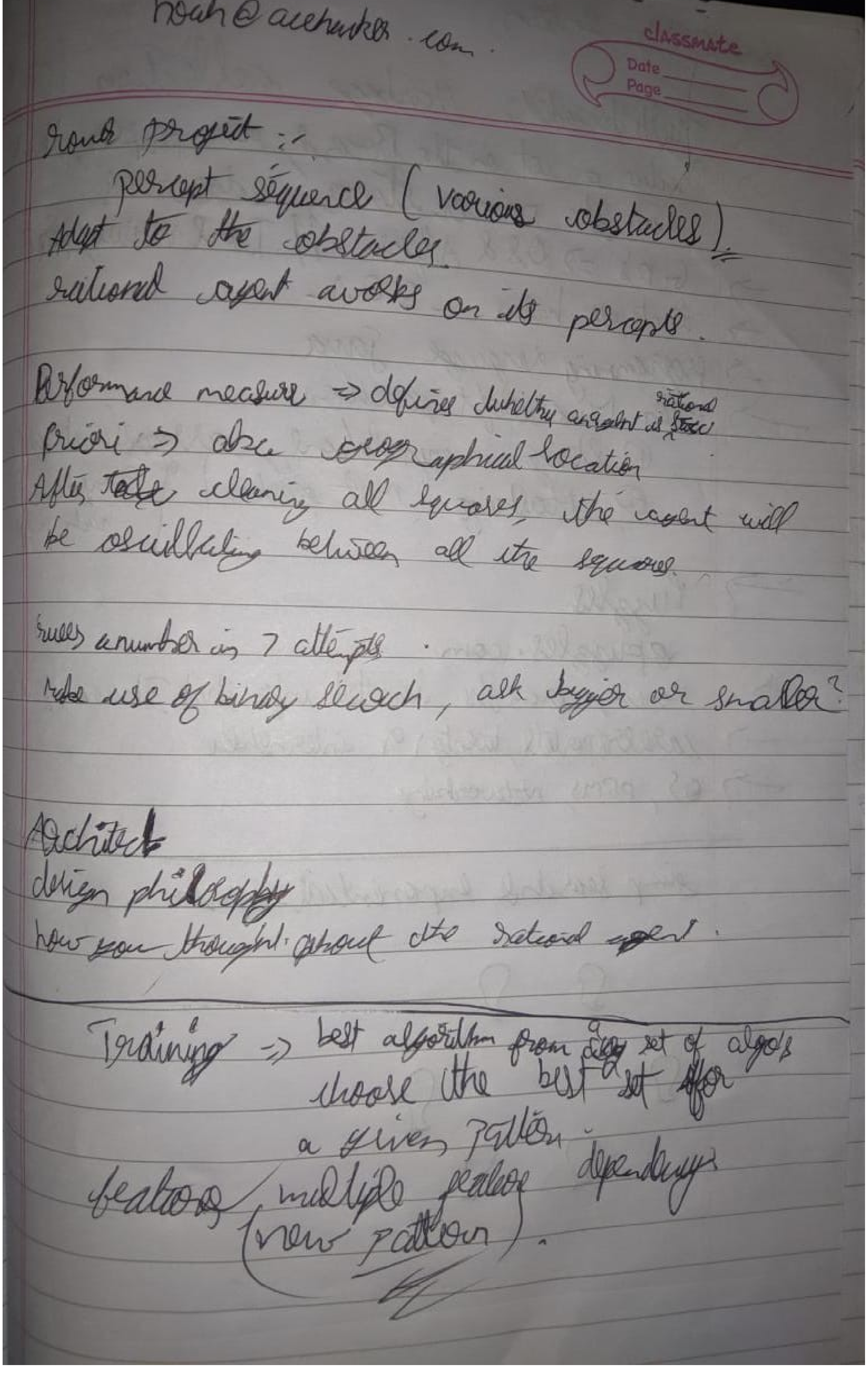
Sensors

Percepts

function

Actions

Actuators



noah@acehacker.com.

classmate

Date

Page

room project :-

percept sequence (various obstacles).

Adapt to the obstacles.

rational agent works on its percepts.

Performance measure ⇒ defines whether an agent is rational

Priori ⇒ also ~~geo~~ geographical location.

After ~~taste~~ cleaning all squares, the agent will be oscillating between all the squares.

Guess a number in 7 attempts.

make use of binary search, ask bigger or smaller?

Architect

design philosophy

how you thought about the rational agent.

Training ⇒ best algorithm from a ~~[illegible]~~ set of algo's

choose the best set ~~after~~ for a given pattern.

features, multiple features dependency (new pattern).