SHAASTRA 2020-21

Online Micro Mouse Maze

- Participants are required to code an autonomous self-contained bot. The following 4 programming languages are supported :

  1) C
  2) C++
  3) Python
  4) JAVA

- The codes will be run on a simulator available . at https://github.com/whitezeta/mms

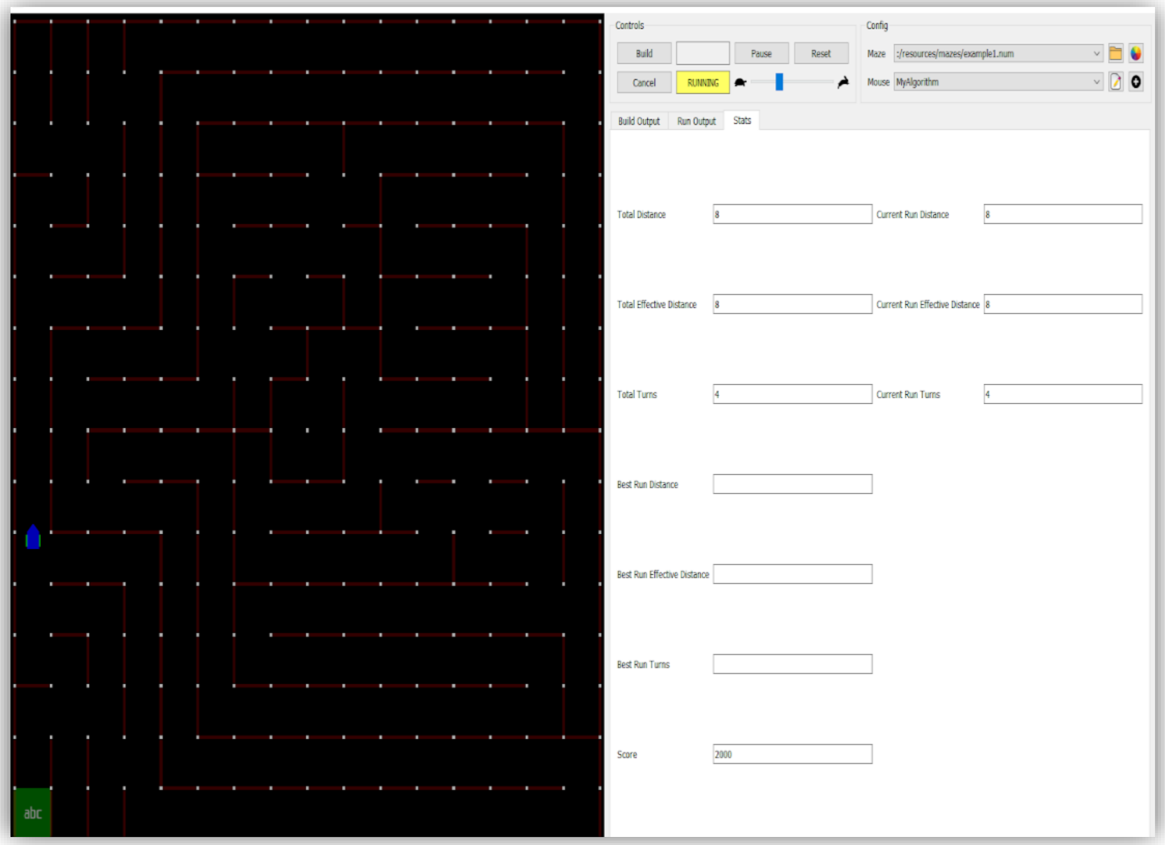
- Micro mouse will be scored based on the score output by the simulator.

- A micro mouse essentially comprises a steering and turning method and sensors to detect the presence or absence of maze walls which has to be specified in the mouse algorithm and control logic to oversee the action of the rest and keep the vehicle 'on track' or to solve the maze.

- The API for the Mouse simulator is clearly demonstrated in the repository itself.

MMS SOFTWARE SPECIFICATIONS:

- Candidates can download the MMS software from the above github link.
- Participants can use this application on their own mazes and test their mouse algorithms and modify their code in consonance with their idea and test its working on the mms application
- Multiple templates of maze files are already available in all the languages in the above link.
- Participants can refer to these files in order to write their code to solve the maze.
- Refer to the README file in the link specified for more details.



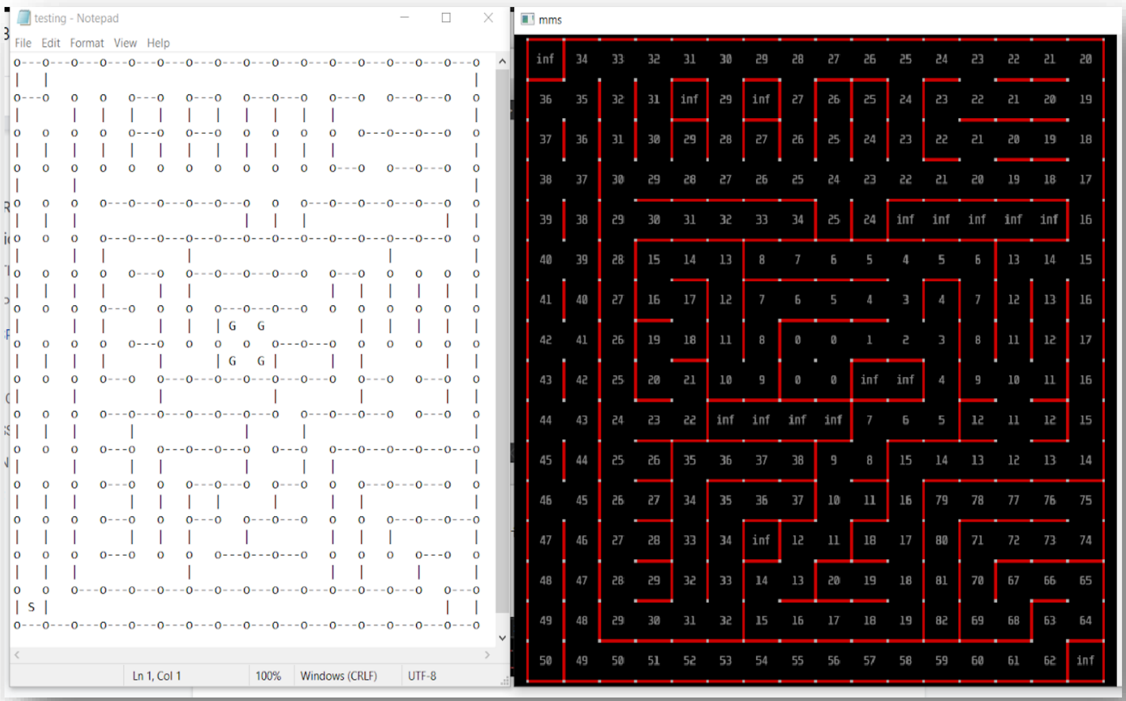
Maze :/resources/mazes/example1.num

Mouse MyAlgorithm

Total Distance	8	Current Run Distance	8
Total Effective Distance	8	Current Run Effective Distance	8
Total Turns	4	Current Run Turns	4
Best Run Distance			
Best Run Effective Distance			
Best Run Turns			
Score	2000		

MAZE SPECIFICATIONS:

- The final custom maze for will be generated using the text files (If interested, refer to the above given link to know how).
- Default dimensions of the maze used in MMS Software will be used in the finals as well. The maze is of size 16 x 16.
- The walls of the maze are represented in red solid lines. The floor of the maze is black. The outermost red wall encloses the entire maze (see the figure below).
- The start square of the maze will be at one of the four corners.
- At the center of the maze shall be a large opening which is composed of 4 unit squares. This central square shall be the destination.
- Small square posts, at the four corners of each unit are called lattice points. The maze shall be constituted such that there is at least one wall touching each lattice point, except for the destination square.



RULES:

Levels	Evaluation Scheme
Level 1	• Participant's code will be run in several different mazes designed by us. • Total score of round 1 will be used as a metric and the top 10 least scored teams will be promoted to Level 2. • In case of having issues like equal score, all the teams with the same score will be promoted to Level 2.
Level 2	• Participant's code will be run in a few different mazes. • Total score of Level 2 will be used as a metric and the top 3 least scored teams will be the winners. • In case of equal scores in Level 2, the total score of Level 1 and 2 will be used as a metric.

- ➢ Participants can clone the template repository of the language of their choice and start coding right away.
- ➢ The start line is at the boundary between the start square and the next unit square. The finish line is at the entrance to the destination square.
- ➢ If a mouse doesn't move or hits the walls of the maze or doesn't reach the destination line before score reaches 2000, they will be given a score of 2000 points.
- ➢ Once the mouse reaches the destination square a pop up would come stating us the total score.

SCORING:

- Winners will be judged based on the least score.

- Speed factor of the mouse has got no credits in scoring as we have control over the simulation (with reference to the application given in github link (https://github.com/whitezeta/mms)

- **Total Score** = 0.2 * (Current run effective distance + Current run turns) + 0.8 * (Total run effective distance + Total run turns)

- Current run effective distance and Current run turns resets whenever the mouse reaches the initial square.

- The sum of total scores from different mazes is then calculated and then given rankings.

SUBMISSIONS:

- Every team has to submit a PDF report consisting of the following details:
  - Build Instructions
  - Run Instructions
  - Team Details (Names, College Details, Phone no, Email Ids of team members, Shaastra ID)

- The team members should make sure that they provide the build and run instructions as accurately as possible.

- You need to make sure that you install the same version of QT as mentioned in the github link.

- The file submitted by the participants must run on the simulator. Make sure it runs on the simulator on your systems before submitting.

- The Code (can be multiple files zipped/tarred together) and the PDF report should be submitted in a google form provided on the Shaastra Website.

TIMELINE:

LAST DATE OF SUBMISSION	25-02-2021 11:59PM
DATE OF RESULTS DECLARATION	28 February, 2021

TEAM RULES:

- Participants can form teams of up to 5 members and no participant can be a member of more than one team.
- Every team member has to register online on the official website for the competition.
- The code may be checked before the evaluation and will be disqualified if an instance of plagiarism is detected.
- The organizers reserve the rights to change any of the above rules as they deem fit. Change in rules, if any, will be notified to the registered participants.
- Winners will be contacted by us shortly once the results are out.

ALL THE BEST