Competition: MicroMouse **Organizer:** Alfredo Weitzenfeld PhD
Date: 9:30 - 11:00 AM May 20th, 2023 **Location:** MSC 2100A

1. Competition Description

Micromouse is an international competition that started in the late 1970s. The objective of the competition is for an autonomous robot (mouse) to learn a maze configuration through multiple navigation trials to find the shortest path from start to goal. The maze comprises 16×16 grid cells each, as shown in Figure 1. As it navigates, the robot needs to keep track of where it is, discover walls as it explores and maps out the maze, and detect when it has reached the goal. Having reached the goal, the robot will typically perform additional searches of the maze until it has found an optimal route from start to finish. Once finished, the robot will run that route in the shortest possible time.

We will use the EPUCK robot and the Webots simulator for this competition.

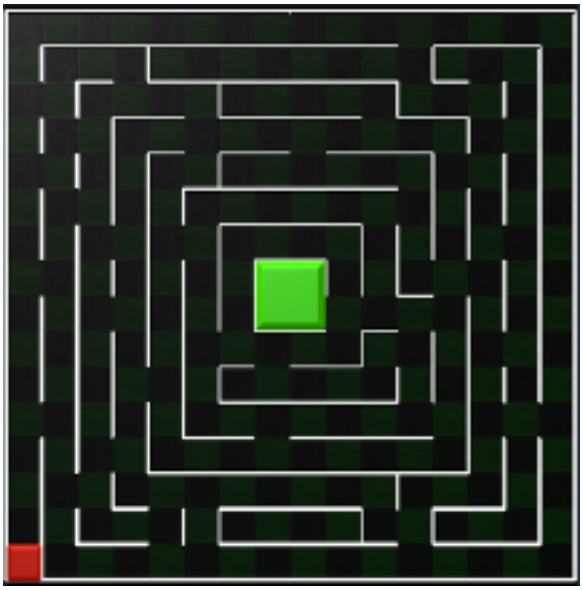

Figure 1. Sample competition maze for simulated robots. The starting location is highlighted in red, and the goal location is highlighted in green.

2. Competition Tasks

The competition requires two separate tasks: (a) map the maze and (b) reach the goal in the shortest time.

2.1. **Rules.** In the following sections, we outline the rules and regulations that will be enforced for this competition.

2.1.1. *Rules for mapping task.*

- This script should navigate the maze and build a data structure that represents the maze.
- This Script can be run up to 10 times. However, each student only has 20 minutes of simulated time for mapping. Note that if the script is reset, the timer stops until the mapping script resumes.
- This script should save the maze data structure using some known method (such as pickle) so that the fastest-run script will be able to utilize the map created during mapping.
- You are to assume that the starting location of the robot is known. However, no additional information should be added during or in between mapping runs.
- The script must report the total number of cells that have been mapped and the total simulated time elapsed to run the mapping algorithm. To get the simulated time, use the Python line $robot.getTime()$. See Webots documentation for Robot object here.

2.1.2. *Rules for fastest run task.*

- This script should use a data structure representing the maze and find the fastest path from the starting location to the goal location. Students can use any search algorithm they wish.
- The robot can only start from the predefined starting locations, always in one lower left corner.
- In order for a run time to be considered, the robot must start in the starting location and stop once a goal location is reached. If the robot does not stop at the goal, the clock does not stop.
- A run time is considered from when the robot starts navigation until the robot stops at the goal location.
- This Script can be run up to 10 times. However, each student only has 10 minutes of simulated time for the fastest run trials. Note that the timer stops if the script is reset until the fastest-run script resumes.

2.1.3. *Rules for Micromouse.*

- A Micromouse cannot jump, fly, climb, cut, or mark the maze's walls.
- Any violation of these rules will result in immediate disqualification.
- A Micromouse should compute the time from its start to its end in both the mapping algorithm task and the fastest run task and print it to the console once the algorithm is finished. This will be compared to the simulator time.
- If too much difference between the calculated time and simulator time run will be disqualified.
- A EPUCK robot has two wheels. Each wheel has a diameter of 1.6 inches (40.64mm) and an axis length is 2.28 inches (57.912mm)
- The EPUCK robot provided includes:
  - 4 X distance sensors positioned in the front, right, rear, and left of the robot
  - 1 X lidar sensor with a 360-degree field of view and a resolution of 1 layer by 360 readings
  - 1 X camera with color recognition enabled facing the front of the robot
  - Must use EPUCK provided. This can be modified with additional sensors cost of 10 points per sensor.

2.1.4. *Rules for Maze.*

- The maze comprises multiples of 18 cm x 18 cm unit cells. The entire maze is composed of 16 x 16 unit cells.
- The maze walls are 5 cm high and 1.2 cm thick. The maze is enclosed by the outside walls.

- The sides of the walls are white.
- The start of the maze is located at one of the four corners. The start cell is bounded on three sides by walls. The destination goal is the four cells at the center of the maze. The goal area has only one entrance.
- Each cell has at least one wall at each corner of a cell.
- Multiple paths to the destination cell are guaranteed. The destination will be positioned so that a wall-following mouse cannot find it.
- The Competition mazes will not be seen until evaluation time

2.1.5. *Evaluation Criteria.* The mapping algorithm accounts for 60% of the final score. We will use Equation 1 to calculate the mapping algorithm's score.

$$m = \frac{90c}{256}\sqrt{\frac{1500 - t}{1500}} + \frac{10c}{256} \tag{1}$$

Where m is the score weight, t is the time the robot takes to perform mapping, and c is the number of cells mapped by the algorithm.

The fastest-run algorithm will account for 40% of the final score. We will use Equation 2 to calculate the fastest run algorithm's score.

$$n = \frac{-1}{6}t + 100 \tag{2}$$

Where t is the robot's time navigating from the starting location to the goal.

The final score f will be calculated by using a weighted average of the mapping score m and the fastest run score n. This is defined in Equation 3.

$$f = .6m + .4n \tag{3}$$

The final score f will determine the competition's winner.

3. Starting Code and Supporting Material

To get started, we have put together a code base that includes everything needed to get started using Webots and developing your solutions. This code base includes two sample mazes that adhere to the rules outlined above; an EPUCK robot with all the previously mentioned sensors and a motion library to help make the robot carry out basic motions.

The code base and supporting materials can be found on our GitHub page. The link is provided here. If you have any questions or concerns, please contact Chance Hamilton at chamilton4@usf.edu.