WGU

COMPUTER SCIENCE CAPSTONE – C964

Vu Dao

Student ID #000861887

June 2024

Part A: Letter of Transmittal

June 11, 2024

Mr. John Doe

WGU Robotic

4001 S 700 E #300

Millcreek, UT 84107

Dear Mr. John Doe,

I am pleased to submit the proposal for the development of a new navigation system for Bubble Rob, our innovative search and rescue robot. This proposal outlines the limitations of the current navigation system and proposes a significant improvement through the implementation of a Q-Learning based AI algorithm.

The new navigation system aims to address the challenge of consistently reaching all targets within complex maze-like environments. By leveraging Q-Learning, Bubble Rob will learn and adapt its navigation strategy in real-time, leading to a substantial increase in its effectiveness and efficiency during missions.

The development of a Q-Learning based navigation system for Bubble Rob will have a positive impact on various stakeholders such as: enhancing reputation for innovation in robotics technology and increasing competitiveness in the search and rescue robot market.

The proposal details the project's objectives, methodology (using the Waterfall approach), and expected benefits. Additionally, it includes an estimated 100 hours of developer work at a rate of $100 per hour.

The selected developer is a highly regarded robotics engineer with a passion for artificial intelligence and its applications in real-world robotics. The developer holds a Ph.D. in Robotics Engineering and possesses extensive experience in developing and implementing AI algorithms for robot navigation and control.

It is my pleasure to discuss this proposal with you further and answer any questions you may have. Please do not hesitate to contact me at vd@wgu.edu.

Thank you for your time and consideration.

Sincerely,

VD

Part B: Project Proposal

Project Summary

WGU Robotic has developed a promising robot called "Bubble Rob" designed for navigating and performing tasks within complex maze-like environments. Initial testing demonstrates Bubble Rob's ability to move and search for targets within these environments. However, the current navigation system utilizes a simple algorithm that proves insufficient for consistently reaching all targets. This limitation hinders Bubble Rob's effectiveness in real-world applications.

To overcome this challenge, WGU Robotic proposes a redesign of Bubble Rob's navigation system using an Artificial Intelligence (AI) approach specifically, a reinforcement learning-based algorithm. This new approach aims to equip Bubble Rob with the ability to learn and adapt its navigation strategy in real-time, leading to a significant improvement in its ability to locate and reach all targets within a complex environment.

The new AI algorithm will enable Bubble Rob to learn and adapt its navigation strategy within complex environments. This will lead to a substantial increase in its success rate of reaching all designated targets, improving overall mission effectiveness.

Algorithm – Environment - Data Summary

Algorithm

This project will utilize Q-Learning, a reinforcement learning technique, to develop Bubble Rob's new navigation system. Q-Learning employs a data structure called a Q-Table.

Q-Table: Imagine a spreadsheet-like structure with rows representing each possible state (e.g., robot's location within the maze) and columns representing each available action (e.g., move up, down, left, right). Each cell in the Q-Table holds a Q-value, which signifies the expected future reward for taking a specific action within a given state.

The Q-Learning algorithm will iteratively update the Q-values based on Bubble Rob's experiences within the maze environment. As the robot explores and interacts with the environment, the Q-values will be adjusted to reflect the effectiveness of each action in different situations. Over time, the Q-Table will converge, enabling Bubble Rob to select the actions with the highest expected future rewards, leading it to navigate efficiently and locate targets successfully.

Environment

To efficiently test and refine the new navigation algorithm, a dedicated 2D maze testing environment will be built within the project using Python. This environment will offer several advantages:

- Compared to a complex 3D simulation, a 2D maze allows for faster development, testing, and debugging of the algorithm.

- The 2D maze will resemble a chessboard with adjustable width and height, enabling the creation of mazes with varying degrees of complexity. The maze can be generated automatically and programmatically after receiving user inputs such as: seed, perfect score, width, height.
- Each cell within the maze will connect to a maximum of four adjacent cells (up, down, left, right). Walls will be placed to represent blocked connections.
- To simulate real-world scenarios with limited information, the environment will incorporate a "fog layer." This fog will initially obscure unexplored areas, restricting the robot's view to its current cell and adjacent (unblocked) cells.

Data Precautions

This project involving Bubble Rob's navigation system development operates within a controlled environment and focuses on purely technical aspects of AI and robotics. There's no collection or use of real-world data, and Bubble Rob won't be interacting with people or situations that raise ethical or legal concerns. This allows us to concentrate our efforts on the technical challenges of designing and testing the Q-Learning algorithm and the 2D maze environment.

Implementation

This project will utilize the Waterfall methodology, a well-defined, sequential approach to software development. Here's a breakdown of the development phases:

1. Requirements Gathering:
    - Define the specific functionalities and performance metrics for the Q-Learning algorithm and the 2D maze environment.
2. Design:
    - Design the architecture of the Q-Learning algorithm, including the Q-Table structure and the logic for updating Q-values.
    - Develop the detailed design of the 2D maze environment, outlining the functionalities for maze creation, robot movement simulation, and interaction with the fog layer.
3. Implementation:
    - Develop the Q-Learning algorithm in Python, implementing the designed logic for state representation, action selection, and Q-value updates.
    - Build the 2D maze environment application in Python, incorporating functionalities for maze generation, robot movement simulation, and the fog layer.
4. Verification:
    - Conduct testing of the Q-Learning algorithm within the 2D maze environment to ensure it functions as designed.
5. Maintenance:
    - Establish a maintenance plan for addressing any bugs or incorporating future improvements to the navigation system.

Timeline

Milestone or deliverable	Duration (hours or days)	Projected start date	Anticipated end date
Design & Planning	2 days	6/11/2024	6/13/2024
Develop & Test 2-D maze class	2 days	6/13/2024	6/15/2024
Develop & Test robot class	2 days	6/15/2024	6/17/2024
Develop & Test Q-model algorithm	2 days	6/17/2024	6/19/2024
Final Test & Deploy	2 days	6/19/2024	6/21/2024
Monitoring & Maintenance	Ongoing	6/21/2024	Ongoing

Evaluation Plan

Verification

- Conduct code reviews for the Q-Learning algorithm and the 2D maze environment application to identify and rectify any errors or logical inconsistencies.
- Perform unit testing of individual components of the code (e.g., Q-value update function, maze generation module) to verify their functionality in isolation.
- Conduct integration testing to ensure all components of the system (Q-Learning algorithm, maze environment, testing framework) work seamlessly together.
- Perform system testing in various maze configurations with different target placements and fog layer settings to verify the overall functionality of the navigation system.

Validation

Compare the performance of Bubble Rob with the new Q-Learning navigation system against its performance with the current simple navigation system. This will involve metrics like:

- Success rate in reaching all targets within the maze.
- Average time taken to complete a maze traversal.

Resources and Costs

This project is estimated to require funding for approximately 100 hours of developer work. Assuming an hourly rate of $100 for the developer, the total cost of personnel would be:

- Developer Cost = $100 * 100 hours = $10,000

In addition to personnel costs, there might be some minor expenses for software licenses or compute resources. However, these are typically minimal for this type of project.

Part C: Application

Github

https://github.com/vdao5/WGU-CS-Capstone-C964

Files

env
| --- common.py
| --- maze.py
| --- robot.py
| --- qtable.py
test_maze.py
test_model.py
test_robot_1.py
test_robot_2.py

c951-BubbleRob-lua.ttt
Essay.pdf

Part D: Post-implementation Report

Solution Summary

The initial navigation system for WGU Robotic's search and rescue robot, Bubble Rob, utilized a simple algorithm that proved insufficient for consistently reaching all targets within complex maze-like environments. This limitation hindered Bubble Rob's effectiveness in real-world scenarios.

The implemented project addressed this challenge by developing a new navigation system for Bubble Rob based on Q-Learning, a reinforcement learning technique. This involved:

- A Q-Learning algorithm was designed and implemented using Python. This algorithm enables Bubble Rob to learn and adapt its navigation strategy based on past experiences within the environment.
- A dedicated 2D maze testing environment was built within Python to efficiently train and evaluate the Q-Learning algorithm. This environment allowed for customization of maze complexity and the incorporation of a "fog layer" to simulate real-world scenarios with limited information.

The Q-Learning algorithm allows Bubble Rob to learn and adapt its navigation strategy in real-time, leading to a significant improvement in its ability to locate and reach all targets within a complex environment.

Environment Summary

To efficiently train and evaluate the Q-Learning algorithm for Bubble Rob's navigation system, a dedicated 2D maze testing environment was developed within the project. This environment offered several key functionalities:

Customization: The maze environment was designed using a 2D grid structure, resembling a chessboard. This allowed for customization of the maze's width and height, enabling the creation of mazes with varying degrees of complexity to challenge the Q-Learning algorithm. A random maze can be programmatically created by using the generate function included in the class library.

Cell Connections: Each cell within the maze could connect to a maximum of four adjacent cells (up, down, left, right). Walls were programmatically placed to represent blocked connections, defining the layout and pathways within the maze.

Target Placement: The environment allowed for the designation of target locations within the maze. These targets represent the locations Bubble Rob needs to navigate to during training and evaluation.

Fog Layer Simulation: To simulate real-world scenarios where Bubble Rob might have limited information about its surroundings, the environment incorporated a "fog layer" functionality. This fog layer initially obscured all unexplored areas of the maze. As Bubble Rob explored its surroundings, only its current cell and potentially visible adjacent cells (depending on wall placement) would be revealed, limiting its view, and mimicking limited information conditions.

Machine Learning

The core of the new navigation system for Bubble Rob is a machine learning algorithm based on Q-Learning, a reinforcement learning technique. Here's a breakdown of the implementation details:

1. Q-Table Design:

   A Q-Table, a fundamental data structure in Q-Learning, was implemented. This table represents the robot's knowledge of the environment. It consists of:

   - States: Each row represents a state that Bubble Rob can encounter within the maze environment. A state could be defined as Bubble Rob's location within the maze (e.g., coordinates within the 2D grid) along with any relevant information about its surroundings (e.g., presence of a wall, fog layer visibility).
   - Actions: Each column represents a possible action that Bubble Rob can take in each state. This typically includes actions like moving up, down, left, or right within the maze.
   - Q-Values: Each cell in the Q-Table holds a Q-value, which signifies the expected future reward for taking a specific action (moving in a particular direction) within a given state (current location and surroundings).

2. Learning Process:

   The Q-Learning algorithm employed a process of exploration and exploitation to learn and update the Q-values within the Q-Table:

- Exploration: During training within the 2D maze environment, Bubble Rob initially explored the maze by taking random actions. As it interacted with the environment (making good / bad moves, reaching targets), it received rewards or penalties based on its progress.
- Exploitation: Based on the received rewards and penalties, the Q-values in the Q-Table were updated. This update process considered both the immediate reward for the action and the potential future rewards achievable from subsequent actions. Over time, the Q-Table converged, with higher Q-values associated with actions that led to successful target acquisition and efficient maze exploration.

3. Action Selection:

   Once trained, the Q-Learning algorithm used the Q-Table to select the most promising action for Bubble Rob to take within a given state during real-time navigation. The algorithm would typically:

   - Prioritize High Q-Values: Preferentially select actions with the highest Q-values in the Q-Table, as these represent the actions with the greatest expected future reward for reaching targets efficiently.
   - Balance Exploration and Exploitation: Incorporate a small degree of randomness (exploration_rate) in action selection to allow for continued exploration and potential discovery of even better strategies.

The successful implementation of the Q-Learning algorithm has equipped Bubble Rob with a powerful tool for navigating complex environments and achieving its search and rescue goals.

Validation

Convergence Rate: This metric measured the percentage of convergence check during model training. It also represents Bubble Rob's ability to find paths from / to a designated location to / from any location within a complex maze. In real time, we do not need to wait the model to complete the training process by itself when the convergence reached 100%. Instead, a stop would be placed whenever it found the paths we need.

Success Rate: This is the percentage of trials where Bubble Rob successfully reached all designated targets within the maze environment. This metric must be 100% under any test cases.

Completion Time: This metric recorded the time taken by the model to complete a training session as well as by the robot to complete its mission.

Visualizations

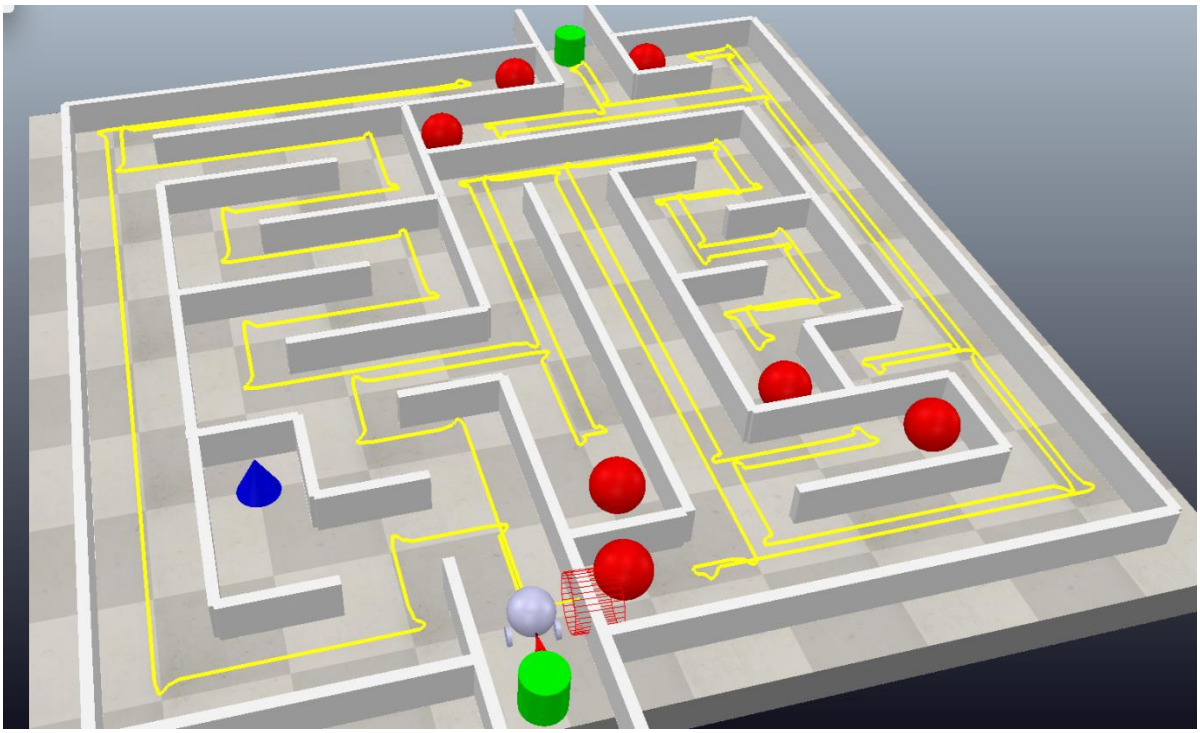

Figure 1: Old navigation system (Failed to reach the cone)

(CoppeliaSim script included in source)

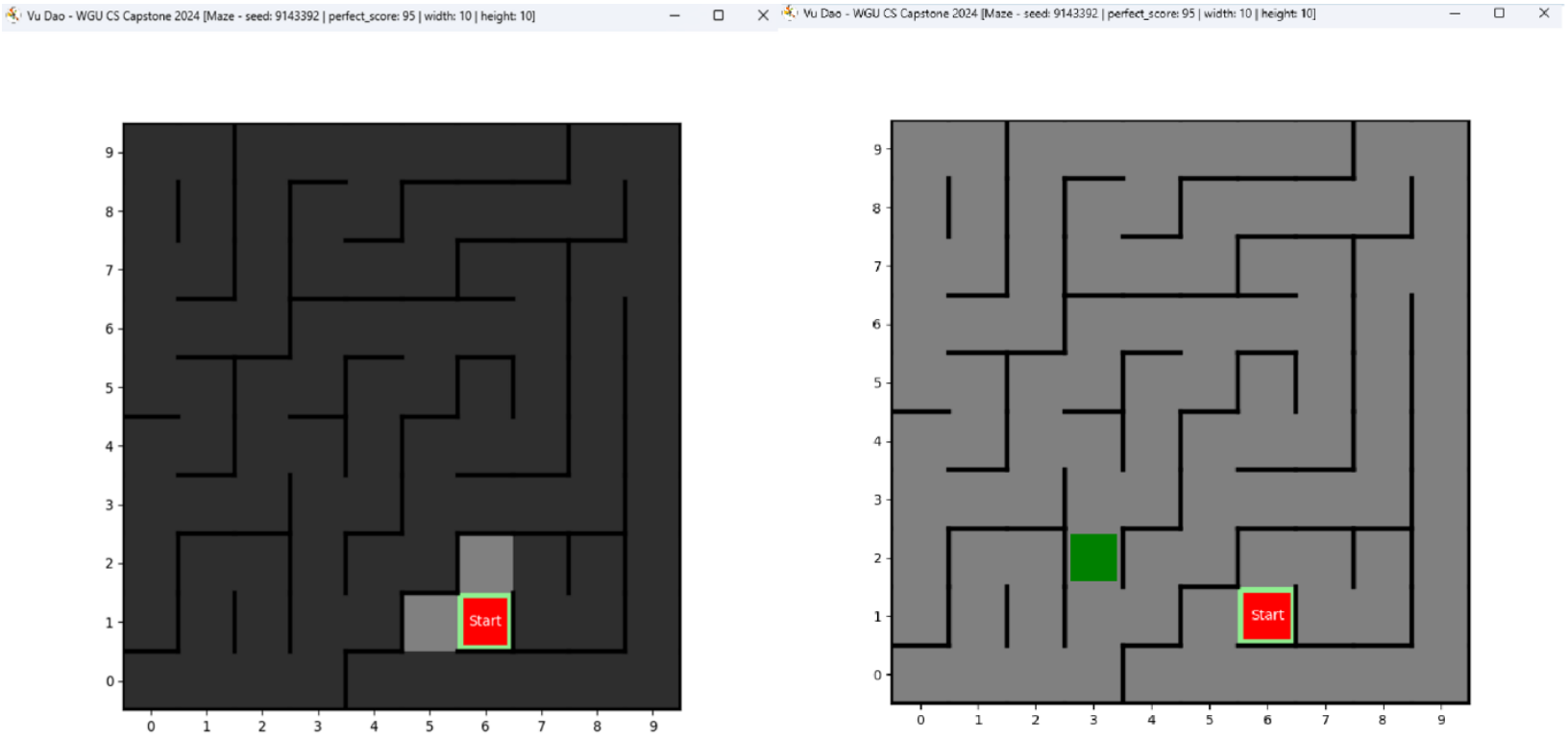

Figure 2: test_maze.py (Test 2-D maze generate function - fog: ON / OFF)

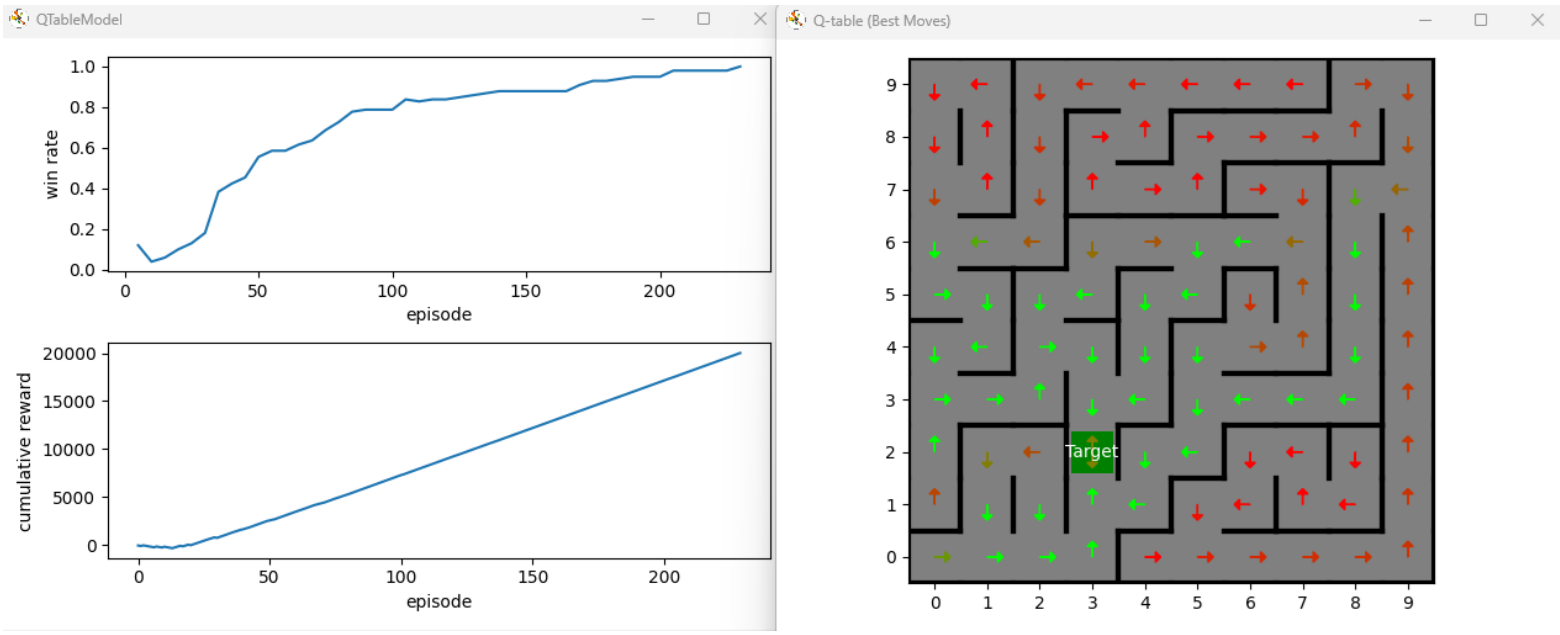

Figure 3: test_model.py (Model Stats & Best Q-Moves)

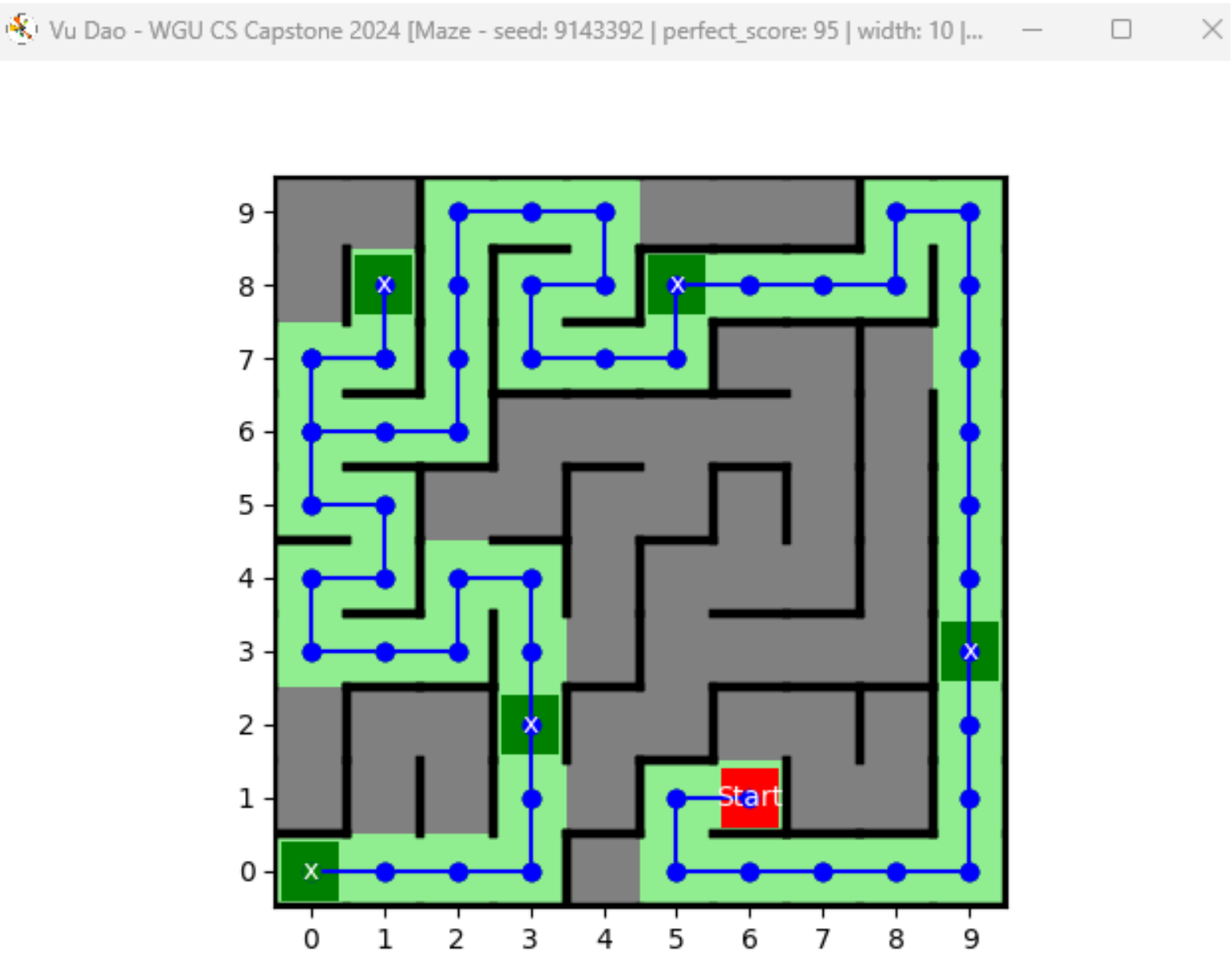

Figure 4: test_robot_1.py (5 targets – fog: OFF – Searching in known environment)

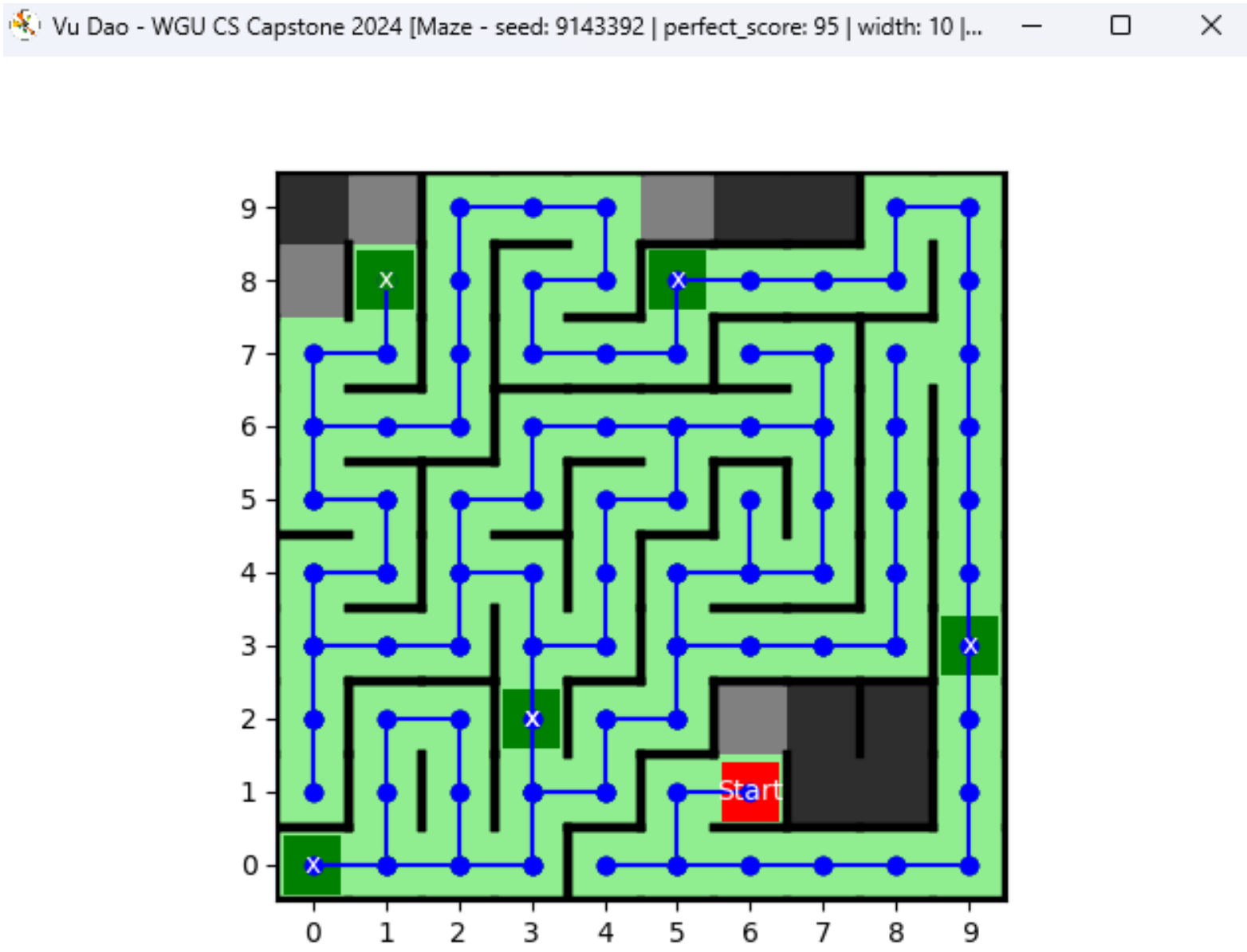

Figure 5: test_robot_2.py (5 targets – fog: ON – Searching in unknown environment)

User's Guide (Windows)

1. Install Python 3 (recommend version > 3.12) – if not installed
    - Check Python version: Windows Command Prompt - "python --version"
    - https://www.python.org/downloads/
2. Install required packages – if missing
    - Open Windows Command Prompt
    - Ensure pip can be run: "python -m pip --version"
    - Install / update pip: "python -m pip install --upgrade pip setuptools wheel"
    - Install numpy package: "python -m pip install numpy"
    - Install matplotlib package: "python -m pip install matplotlib"
3. Open submitted zip file or download at https://github.com/vdao5/WGU-CS-Capstone-C964
4. Extract source files to a project folder. There are 4 test files in the project folder:
    - "test_maze.py"
        - Test environment
        - Manual control using arrow keys
        - fog = True / False
        - Maze.generate(seed = any number, perfect_score = [0, …, 100], w > 0, h > 0)
    - "test_model.py"
        - Test QModel training
        - Plot training stats
        - Plot best Q-moves
        - Test moves from every cells

    - model.train(…): (see document – qtable.py > class QModel > train(…))
- “test_robot_1.py”
    - Test Navigation System 1
    - Task: Searching – fog = False
- “test_robot_2.py”
    - Test Navigation System 2
    - Task: Searching – fog = True

5. Run test file.
    - Install Visual Studio Code (or alternatives)
        - https://code.visualstudio.com/download
    - Open the project folder in Visual Studio Code
        - Right click on the folder then select “Open with Code”
    - Open Visual Studio Code terminal
        - Menu > Terminal > New Terminal
    - In terminal, enter “python filename.py”

Reference Page

(None)