Underwater ROV Fleet Management Simulator

- **Mazen Eldeeb (team leader)**
- **Asmaa Aboushady**
- **Kenzy Mohamed**
- **Youssef Mahmoud**
- **Perihane Hossam**

9/25/2024

This project simulates the operation of a fleet of Remote Operated Vehicles (ROVs) designed for various underwater missions, using Python's object-oriented programming and multithreading capabilities. The project consists of a central control system (MMCS) to manage the ROVs and assign missions to them, each vehicle being specialized for different tasks: Exploration, Sampling, or Maintenance.

Key Components:

1. Base Class - `ROV`:

- This is an abstract base class representing a generic ROV. It has attributes like:

  - `rov_id`: Unique identifier for each ROV.

  - `rov_type`: Type of the ROV (e.g., Exploration, Sampling, Maintenance).

  - `status`: Current status (Idle, In Mission, Low Battery, etc.).

  - `battery_level`: Current battery percentage.

  - `mission_queue`: A queue that stores the assigned missions.

  - `position`: Current coordinates of the ROV.

  - `lock`: A threading lock to synchronize status updates in multithreaded execution.

- Methods:

  - `receive_mission`: Adds a new mission to the ROV’s mission queue.

- `check_battery`: Checks the battery status, updates if low.
- `navigate_to`: Simulates navigation to a target location.
- `send_status_update`: Sends a status update of the ROV.
- `execute_mission`: Abstract method to be implemented by subclasses.

2. ROV Subclasses:

- `ExplorationROV`: ROV used for exploration missions such as mapping underwater areas.
- `SamplingROV`: ROV specialized in collecting underwater samples.
- `MaintenanceROV`: ROV for performing maintenance tasks, such as repairing underwater structures.
- Each subclass implements the `execute_mission` method differently, simulating specific tasks such as mapping, sampling, or maintenance.

3. Mission Management & Control System (MMCS):

- The `MMCS` class is responsible for managing the fleet of ROVs and assigning missions. Its primary functions are:
  - Initialization: The user defines the fleet's size and the type of each ROV.
  - Mission Generation: Creates a mission with an ID, type (Exploration, Sampling, Maintenance), and a random target location.
  - Mission Assignment: Randomly assigns missions to ROVs based on their type.
  - Simulation Control: Uses multithreading to simulate multiple ROVs performing their missions concurrently.
  - Terminate Simulation: Concludes the simulation and displays a summary of the fleet's status and battery levels after all missions are completed.

4. Simulation Flow:

- Step 1: The user is prompted to define the number of ROVs and missions.

- Step 2: The user assigns a type to each ROV in the fleet.

- Step 3: The MMCS assigns missions to ROVs based on their types.

- Step 4: Each ROV executes its missions using multithreading, performing tasks like exploration, sampling, or maintenance while navigating to various target locations.

- Step 5: After all missions are completed, the simulation terminates, and a final summary is displayed.

Features:

- Multithreading: Each ROV operates independently, performing missions in parallel.

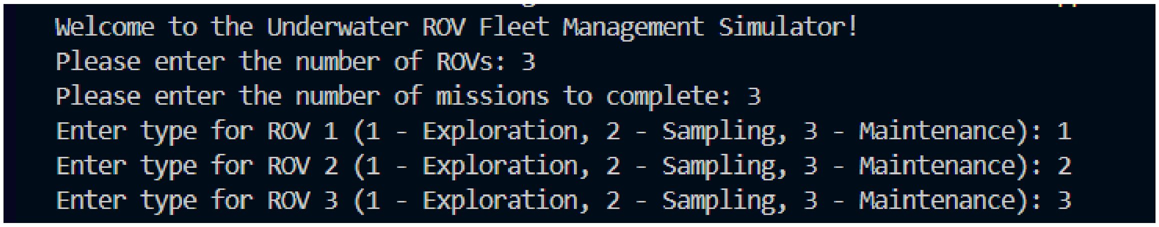
- Mission Queuing: Each ROV can handle multiple missions, executing them sequentially.

- Battery Monitoring: The system tracks battery levels, and ROVs warn when battery levels fall below a certain threshold.

- Dynamic Mission Assignment: Missions are assigned based on the ROV’s specific type and the user’s input.

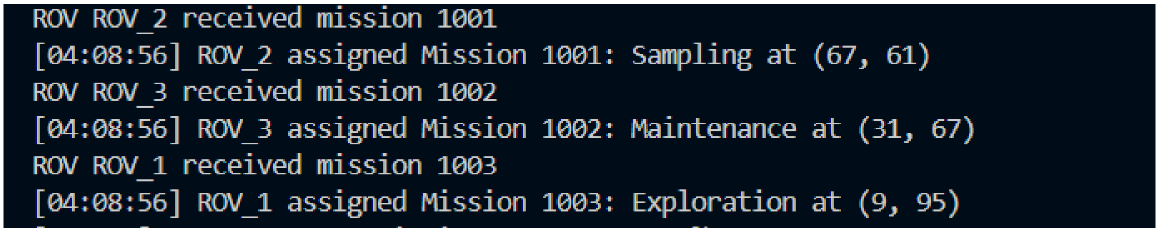
Walk throw the program:

Frist the user choose the number of rov’s that he wants and the type then he chooses the number of missions



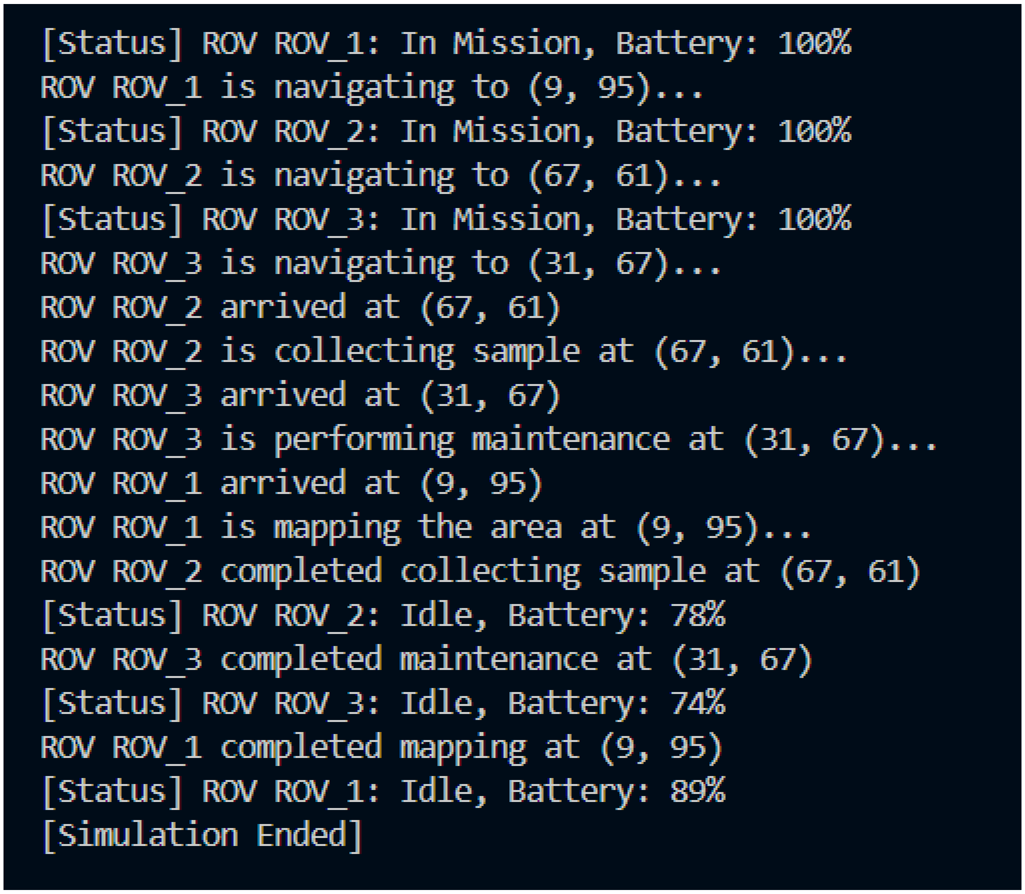
```
Welcome to the Underwater ROV Fleet Management Simulator!
Please enter the number of ROVs: 3
Please enter the number of missions to complete: 3
Enter type for ROV 1 (1 - Exploration, 2 - Sampling, 3 - Maintenance): 1
Enter type for ROV 2 (1 - Exploration, 2 - Sampling, 3 - Maintenance): 2
Enter type for ROV 3 (1 - Exploration, 2 - Sampling, 3 - Maintenance): 3
```

Then each rov is assigned a mission



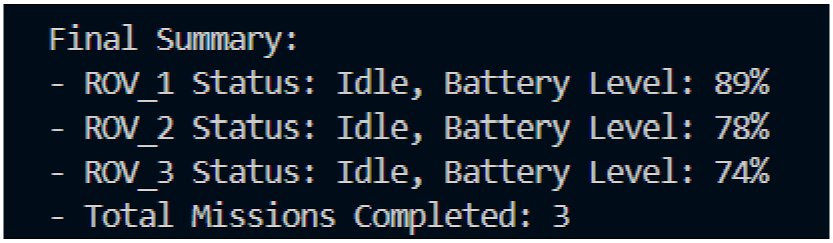
```
ROV ROV_2 received mission 1001
[04:08:56] ROV_2 assigned Mission 1001: Sampling at (67, 61)
ROV ROV_3 received mission 1002
[04:08:56] ROV_3 assigned Mission 1002: Maintenance at (31, 67)
ROV ROV_1 received mission 1003
[04:08:56] ROV_1 assigned Mission 1003: Exploration at (9, 95)
```

Then the misson start and give the user continues uptade on the missions



```
[Status] ROV ROV_1: In Mission, Battery: 100%
ROV ROV_1 is navigating to (9, 95)...
[Status] ROV ROV_2: In Mission, Battery: 100%
ROV ROV_2 is navigating to (67, 61)...
[Status] ROV ROV_3: In Mission, Battery: 100%
ROV ROV_3 is navigating to (31, 67)...
ROV ROV_2 arrived at (67, 61)
ROV ROV_2 is collecting sample at (67, 61)...
ROV ROV_3 arrived at (31, 67)
ROV ROV_3 is performing maintenance at (31, 67)...
ROV ROV_1 arrived at (9, 95)
ROV ROV_1 is mapping the area at (9, 95)...
ROV ROV_2 completed collecting sample at (67, 61)
[Status] ROV ROV_2: Idle, Battery: 78%
ROV ROV_3 completed maintenance at (31, 67)
[Status] ROV ROV_3: Idle, Battery: 74%
ROV ROV_1 completed mapping at (9, 95)
[Status] ROV ROV_1: Idle, Battery: 89%
[Simulation Ended]
```

After the simulation is done the program gives the user a final summary



```
Final Summary:
- ROV_1 Status: Idle, Battery Level: 89%
- ROV_2 Status: Idle, Battery Level: 78%
- ROV_3 Status: Idle, Battery Level: 74%
- Total Missions Completed: 3
```

Task division

- Mazen el deep: Base class (ROV)
- Asmaa abo shady : main class (MMCS)
- perihane Hossam :subclass (ExplorationROV) & presentation
- kenzy Mohamed :subclass (Sampling ROV)
- Youssef Mahmoud: subclass (MaintenanceROV) & documentation

9/25/2024