St Mary's University Twickenham London

AI Project: Techburg Simulation

Author	**: 2412324**
Course Title	**: BSc Computer Science**
Module Convenor	**: Prins Butt**
Date	**: 1/16/2025**

Abstract

In this project, a dynamic and interactive program has been designed with interactions between survivor bots, scavenger swarms, malfunctioning drones, spare parts, and recharge stations within a 40x40 grid environment. According to the given task, the simulation explores different behaviours where survivor bots collect spare parts and recharge energy to stay alive while malfunctioning drones consume spare parts and destroy survivor bots if needed. Scavenger swarms also consume depleting spare parts and emit a decay effect that reduces energy levels of nearby entities, and they even multiply after consuming ten decaying spare parts. There are five recharge stations in the simulation whose location is fixed, and each station can hold five survivor bots at a time. They consume restored parts to restore energy, and when multiple survivor bots occupy the same station at a time, the bots share information about the whereabouts of spare parts.

Table of Contents

Table of Figures

Introduction

This project simulates a hypothetical apocalyptic city named Techburg, where robots, including bots, drones, scavenger swarms, and recharge stations, are the only inhabitants. The ecosystem revolves around survivor bots collecting spare parts to recharge, survive, and multiply while drones and scavengers aim to disrupt their efforts. Drones' only objective is to destroy the survivor bots, while scavengers compete for spare parts to replicate and consume resources, emit decay, and weaken nearby entities. Each element has defined roles and objectives that contribute to the simulation's main goal: to observe the interactions within this robotic ecosystem and analyse how long it can sustain itself under defined conditions.

System Design

There are multiple objectives for this project, and fulfilling them has created this ecosystem of a 40x40 2D grid with survivor bots, drones, scavenger swarms, spare parts, and recharge stations. Each entity has its own specific requirements, and building these entities based on these requirements is the main aim of this section. The simulation ends automatically when there are no spare parts or survivor bots. The requirements of the entities with their mechanics and interactions are listed below:

i. **Spare parts:** In this, the spare parts have to be spread out randomly throughout the map. It has not been defined how many spare parts must be there; as such, for this project, twenty-five spare parts have been spread out. Other characteristics of the spare parts are:
   - It depletes by 0.1% every second in the simulation.
   - It can increase the energy of bots if consumed by (small = 3%, medium = 5%, large = 7%).
   - It does not deplete its energy in the recharge station.
   - It recharges back to maximum capacity in the recharge station by 5% every second.

ii. **Survivor bots:** These survivor bots are randomly spread out through the map. There is no predefined instruction on how many survival bots are needed; as such, for this project, ten bots are initially created. Other characteristics of these bots are displayed below:
   - Each bot's main goal is to search for the spare parts.
   - Each bot can carry only one spare part at a time to the recharge station.
   - Each bot depletes its energy by 5% with every cell it moves.
   - Each bot waits for the spare parts to be 100% in the recharge station and consumes the spare parts.
   - If the energy of the bot is 5% or lower, the bot consumes the spare parts immediately without taking it to the recharge station.
   - If the energy of the bot is 0%, it will be inactive and removed from the simulation if not recharged.
   - Each bot can recharge by 1% every second at the recharge station.

- When more than one bot is present in the recharge station, there’s a 20% chance to create an additional bot (costing 30% energy each) and a 5% chance of creating one more bot (costing 50% energy each), but the chances are very low.
- When the bots consume three spare parts, the speed increases to 100%, which was 80% initially.
- Bots can detect spare parts two cells away.
- The maximum detection range of each bot is 150%, which increases when five spare parts have been consumed, which was initially 100%.

iii. **Drones:** The main characteristics of a malfunctioning drone in this map is to detect and pursue the surviving bots. Below are the other characteristics.

- The energy of the drone depletes by 20% when the drone pursues bots.
- If the energy of the bot is more than 80% when the drone catches it, the drones destroy the bots, causing the bot to drop any spare parts.
- If the energy of the bot is more than 60% when the drone catches it, the drone disables the bot, reducing its energy by 20%, causing the bot to drop any spare parts.
- If the energy of the bot is more than 20% when the drone catches the bot, the drone delivers a shock, making the bot reduce its energy by 5%, causing the bot to drop any spare parts.
- When the drone's energy drops to 20% or less, it enters hibernation mode and recharges at a rate of 10% per simulation until it reaches full power.

iv. **Scavenger swarms:** Scavengers are a cluster of nanobots that roam the map that consumes depleting parts. Initially, two swarms are implemented.

- Scavenger swarm emits decay effect, which causes the bots and drones to lose 3% energy per simulation when near one cell of the swarm.
- Scavengers consume spare parts to boost energy by (small = 1%, medium = 2%, and large = 3%).
- Scavengers can self-replicate when ten spare parts are consumed.

v. **Recharge stations:** The recharge station's main character is to collect and restore the energy of spare parts and bots. The recharge stations also have the following characteristics:

- There are five stations in total: four in each corner of the map and one in the centre.
- Each station can hold five bots at a time.
- When multiple bots occupy the same station, they share information about the whereabouts of spare parts.

System Implementation

The simulation was implemented using Python with Tkinter for GUI purposes and the random library to generate random positions in the map. The key components of the implementation included entity behaviours, a grid system, and visualisation. The tools and libraries used to implement the system are explained below:

i. **Python:** Python was used as a core programming language.
ii. **Tkinter:** Tkinter was used for GUI to render a grid and visualise real-time interactions.
iii. **Random:** To ensure unpredictable interactions between entities, random positions were generated for entities during initialisation using the "random" library.
iv. **Custom Classes:** Each entity and the grid system had its own classes designed accordingly.

Logic for Grid Updates

The 2D grid is 40x40 in size, where entities interact with each other in real time. The map updates every second, according to the movements, interactions, and changes in the entity. Tkinter is used to represent the grid visually. The simulation also displays the log of every activity in the simulation. In the map, each cell is color coded: ·

- Survivor Bots (Green)
- Drones (Orange)
- Scavengers Swarms (Red)
- Spare Parts (Yellow)
- Recharge Stations (Blue)
- Empty Cells (White)

Logic for Entity Placement

The "random" library was used for dynamic and fair distribution of entities such as bots, drones, scavenger swarms, and spare parts. These entities were placed at random positions on the grid using the "random.randint" function, which ensures each simulation is unique. Randomised attempts were also made to locate empty cells for entity placement or multiplication.

Special Placement: Only the recharge stations are placed in a fixed position at the four corners and the centre of the grid to make the simulation easier.

Logic for Task Prioritisation

In this project, each entity has its own priority, and they are all explained below:

i. **Survivor Bots:** Survival is the main priority of bots by collecting spare parts, recharging, and avoiding decay zones. The priority logic of the survivor bot is listed below:
    - If the energy is less than 20% of the bot, heads towards the recharge station to reenergise.

- If the energy is more than 20% of the bot, collection of spare parts is prioritised.
- Move towards the spare part that is within reach.
- The bots avoid cells near to scavenger swarms.
- The bots move randomly when there is no other priority to be performed.

ii. **Scavenger Swarms:** The main priority of scavenger swarms is to emit decay effects harmful to bots, drones, and spare parts. Other priorities are listed below:
   - It prioritises consumption of depleting spare parts.
   - It multiplies when ten spare parts are consumed.
   - If there's no availability of spare parts or any targets, it moves randomly.

iii. **Drones:** The task priority of drones is to pursue bots. The drones perform the task according to the following:
   - It destroys the bots if the energy of the bot is more than 80%.
   - It disables the bots if the energy of the bot is more than 60%, causing it to drop spare parts.
   - It delivers a shock to the bots if the energy of the bot is more than 20%, causing it to drop spare parts.
   - It enters hibernation mode, recharging at 10% per simulation when the energy is equal to or less than 20%.

iv. **Spare Parts:** The logic of the spare part is that it is spread throughout the grid, and each part loses its energy by 0.1% per second.

v. **Recharge Stations:** The logic of the recharge stations is as follows:
   - It replenishes the energy of a bot to maximum capacity by 5% every second.
   - Recharge stations remain fixed at one location and only interact with survivor bots.

Challenges and Resolutions

During the implementation process, faced a lot of challenges but later was able to resolve them, which is explained below:

i. **Entity Placement**: In the beginning, entity placement was difficult with not knowing where to place the said entity. To solve this problem, random placement and collision detection logic were implemented.

ii. **Entity Visibility:** Some of the entities, like drones or scavengers, were not visible in the simulation at first. Hence, later updated grid visualisation logic and ensured proper placement with a colour code for simplicity.

iii. **Scavenger swarm behaviour:** The scavengers did not multiply after consumption of said spare parts, then updated a function for multiplication and handling decay effects.

iv. **Real-Time Decay Effect:** To ensure this task was successful, decay checks during each grid update were introduced to ensure affected entities lost the supposed energy.

v. **Scalability Issues:** Optimised to operate the grid in a 40x40 environment.

vi. **Spare part depletion:** Introduced spare part class with corrosion and depletion logic, integrating it with scavenger swarms and drones.

Results and Analysis

After successful creation of the hypothetical apocalyptic simulation, it could be seen that bots successfully navigated the grid, collected spare parts, and recharged themselves when they came in contact with decay zones. Similarly, scavenger swarms completed their task with the consumption of depleting parts, emitting a decay effect (depleted energy of bots and drones). As for the drones, they outpaced the bots by destroying, delivering shock, and disabling already depleted bots from scavengers. Hence, it could be said that there's only very little chance of survival for bots.

Performance Metrics

In this project, during the performance analysis, it could be seen that bots collected 60% of the spare parts on average, around 30% of the bots lost their energy due to scavenger swarms, and about 20% due to drones.

Failures

Although the creation was successful, failures were unavoidable. The main problem occurred with the bots when there were multiple decay zones overlapping as well as drones pursuing them. The bots need spare parts to survive, but the drones often outpaced the bots due to their faster movement.

Comparison of Parameters

In comparison, it could be seen that simulation with fewer swarms and drones had a higher bot survival rate, while an increased number of swarms and drones led to resource scarcity and rapid energy depletion of bots.

Discussion

The project successfully demonstrates the interactions between survivor bots, scavenger swarms, and drones in a competitive and limited environment. This project highlights the importance of resource management, task prioritisation, and strategic behaviours for survival.

Strengths

The project's main strength is its capacity to represent dynamic interactions between entities in real-time simulation. Each entity performs their task according to their prioritization. Also, the colour-coded entities displayed make it easier to understand how they interact.

Limitations

Although the project was successfully designed, there were some limitations that made the simulation more challenging. The simulation becomes less dynamic due to the fixed positions of recharge stations. Finally, the grid becomes crowded with drones and scavenger swarms, which results in the depletion of spare parts, leading to Iow survival rates of bots.

Improvements

The project can be improved by implementing a machine learning algorithm, which would enable entities like survivor bots to learn from previous interactions and avoid or make smarter decisions. Recharge stations could be made more dynamic, and optimising the performance of the grid updates and entity behaviours would allow the simulation to enhance its scalability and realism.

Real-World Application

The Techburg simulation could train autonomous systems for real-world tasks like disaster recovery, where bots need to prioritise resources. Furthermore, it also helps students or learners to study strategies for decision-making and survival. (William, Michael & Chikwarti, Dileep Kumar. (2024)

Conclusion

In the end, Techburg successfully achieved the given tasks with the help of real-time grid updates and interactive visualisation. The project provides valuable insights into decision-making processes and the impact of cooperative and competitive dynamics among entities.

However, at the same time, the simulation is doomed to fail since there is no balance at all. The number of survivor bots is outnumbered since it takes certain conditions to be met to replicate itself, while replication affects the energy of other bots as well. This results in survivor bots being destroyed in every simulation. Survivor bots are already designed to drop collected spare parts when attacked by drones or scavengers. To improve the simulation, it could be added that even without the interference of drones or scavengers, when bots are inactive from low energy and removed from the simulation, they should be turned into spare parts, acting as a part of the life cycle to create an ecosystem. Another improvement that can be done is to increase the number of bots, since replication is low, but the attackers are too many. The recharge stations also should be dynamic. Future improvements like machine learning could also be implemented to train the entities and make the simulation more advanced.

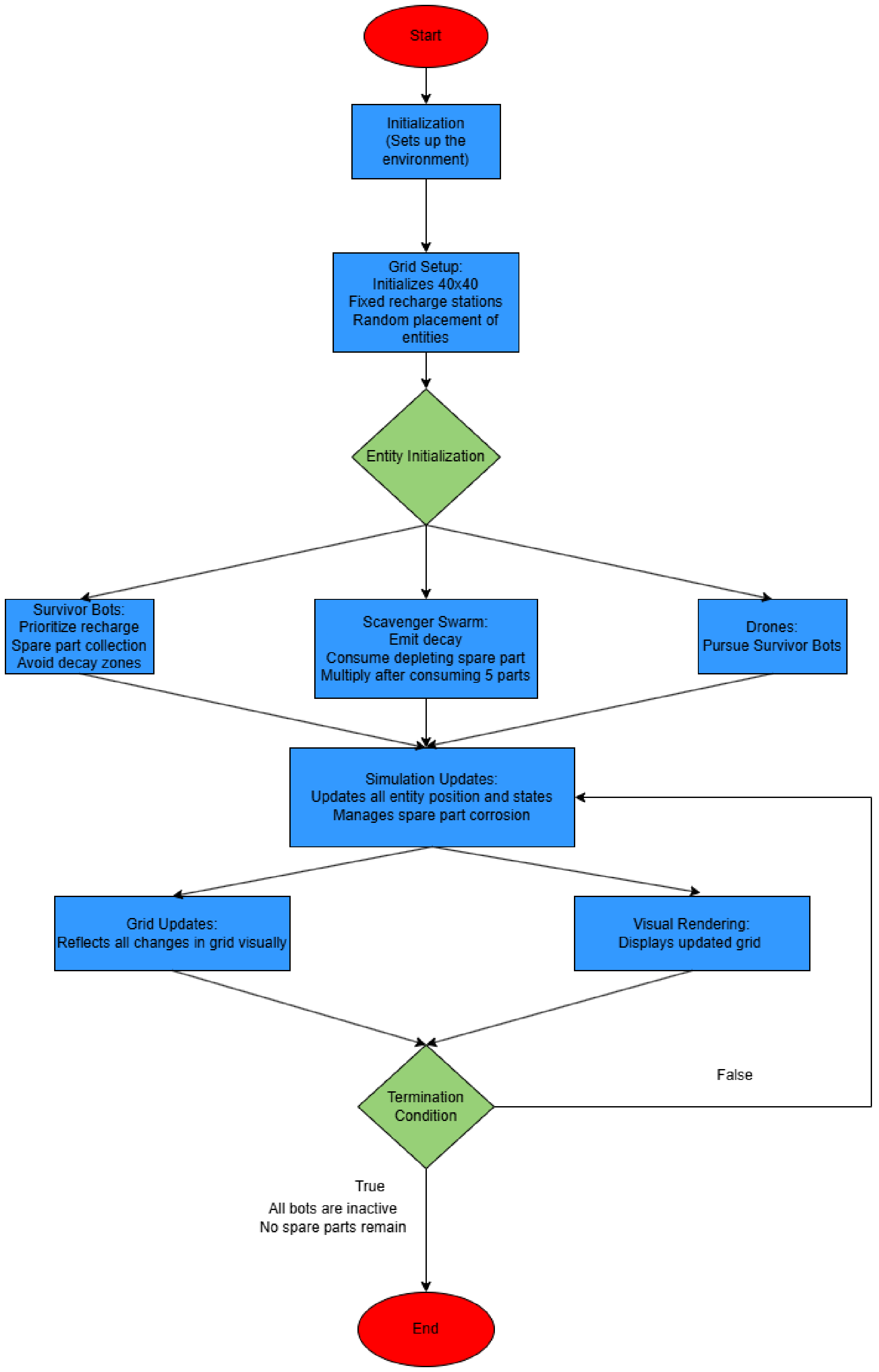
References

Kannan, M.K.Jayanthi. (2024). Python GUI Development using Tkinter Certificate.

Sargent, Robert. (2011). Verification and validation of simulation models. Engineering Management Review, IEEE. 37. 166 - 183. 10.1109/WSC.2010.5679166.

William, Michael & Chikwarti, Dileep Kumar. (2024). Real-World Applications of AI in Adaptive Clinical Trials.

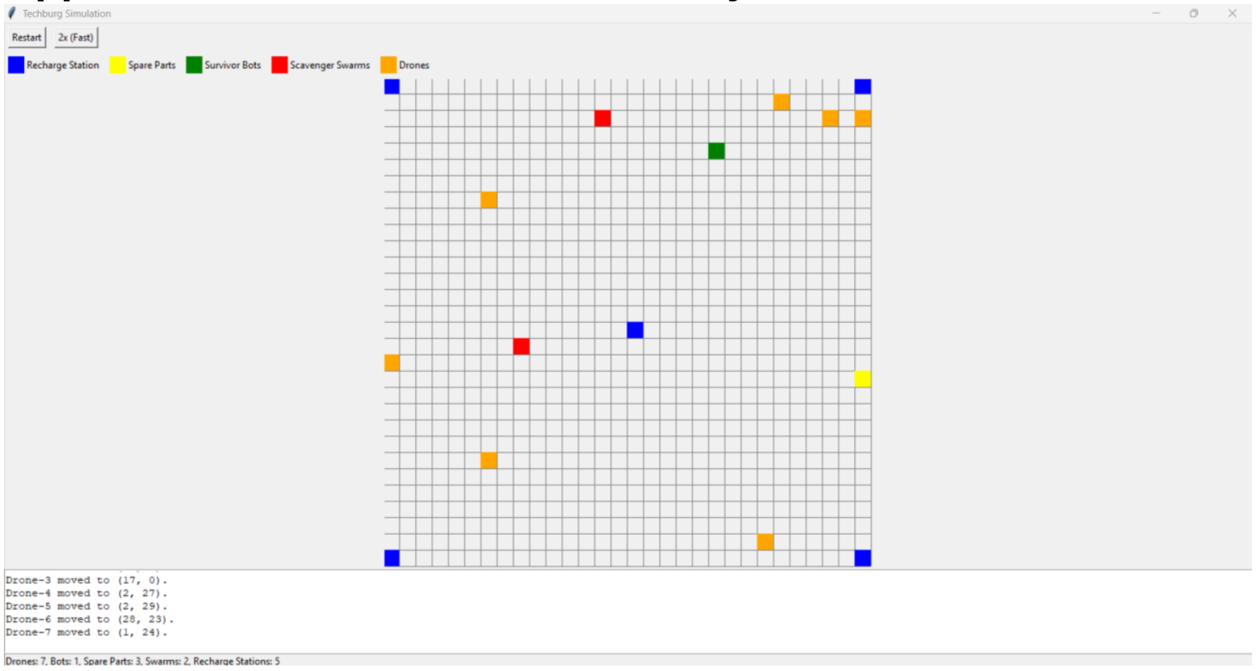
Appendix A: Architectural Diagram



Start
Initialization (Sets up the environment)
Grid Setup: Initializes 40x40 Fixed recharge stations Random placement of entities
Entity Initialization
Survivor Bots: Prioritize recharge Spare part collection Avoid decay zones
Scavenger Swarm: Emit decay Consume depleting spare part Multiply after consuming 5 parts
Drones: Pursue Survivor Bots
Simulation Updates: Updates all entity position and states Manages spare part corrosion
Grid Updates: Reflects all changes in grid visually
Visual Rendering: Displays updated grid
Termination Condition
False
True
All bots are inactive
No spare parts remain
End

Figure 1: Flow Chart

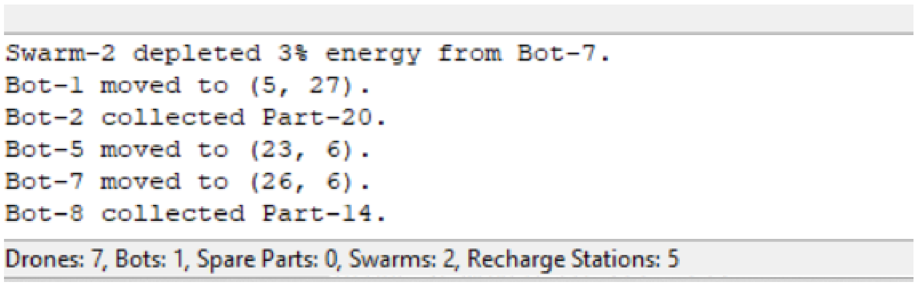
Appendix B: Screenshots of Final System



```
Drone-3 moved to (17, 0).
Drone-4 moved to (2, 27).
Drone-5 moved to (2, 29).
Drone-6 moved to (28, 23).
Drone-7 moved to (1, 24).
```

```
Drones: 7, Bots: 1, Spare Parts: 3, Swarms: 2, Recharge Stations: 5
```

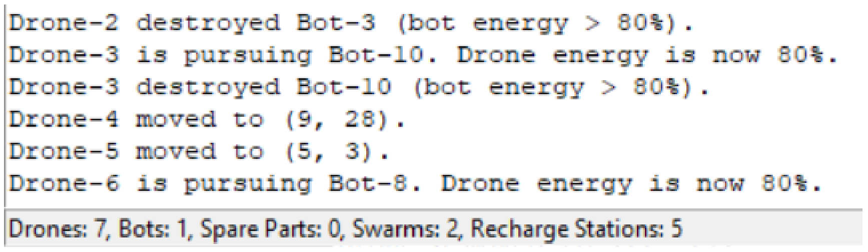
Figure 2: Final system



```
Swarm-2 depleted 3% energy from Bot-7.
Bot-1 moved to (5, 27).
Bot-2 collected Part-20.
Bot-5 moved to (23, 6).
Bot-7 moved to (26, 6).
Bot-8 collected Part-14.
```

```
Drones: 7, Bots: 1, Spare Parts: 0, Swarms: 2, Recharge Stations: 5
```

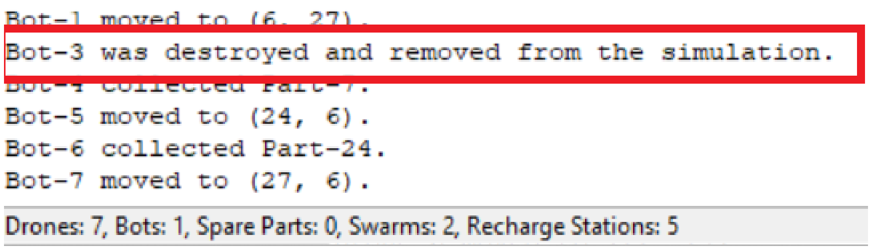
Figure 3: Capturing the performance of swarm and bots



```
Drone-2 destroyed Bot-3 (bot energy > 80%).
Drone-3 is pursuing Bot-10. Drone energy is now 80%.
Drone-3 destroyed Bot-10 (bot energy > 80%).
Drone-4 moved to (9, 28).
Drone-5 moved to (5, 3).
Drone-6 is pursuing Bot-8. Drone energy is now 80%.
```

```
Drones: 7, Bots: 1, Spare Parts: 0, Swarms: 2, Recharge Stations: 5
```

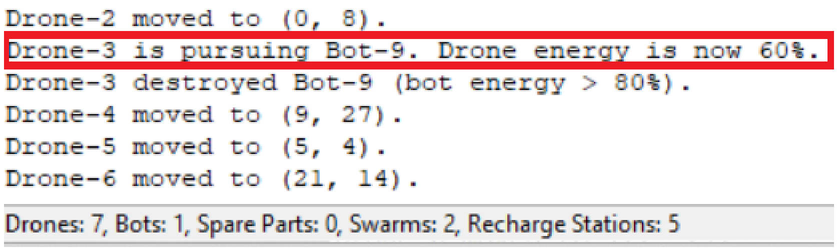
Figure 4: Capturing the performance of drones



```
Bot-1 moved to (6, 27).
Bot-3 was destroyed and removed from the simulation.
Bot-4 collected Part-7.
Bot-5 moved to (24, 6).
Bot-6 collected Part-24.
Bot-7 moved to (27, 6).
```

Drones: 7, Bots: 1, Spare Parts: 0, Swarms: 2, Recharge Stations: 5

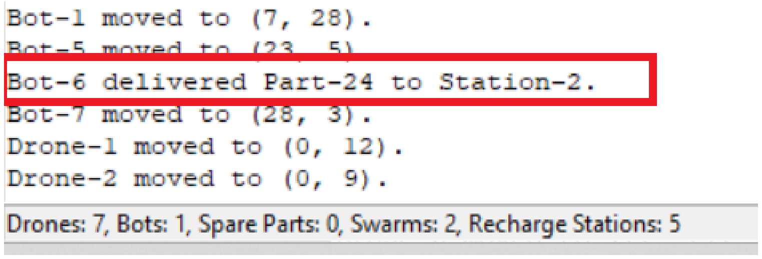
Figure 5: Bot destroyed



```
Drone-2 moved to (0, 8).
Drone-3 is pursuing Bot-9. Drone energy is now 60%.
Drone-3 destroyed Bot-9 (bot energy > 80%).
Drone-4 moved to (9, 27).
Drone-5 moved to (5, 4).
Drone-6 moved to (21, 14).
```

Drones: 7, Bots: 1, Spare Parts: 0, Swarms: 2, Recharge Stations: 5

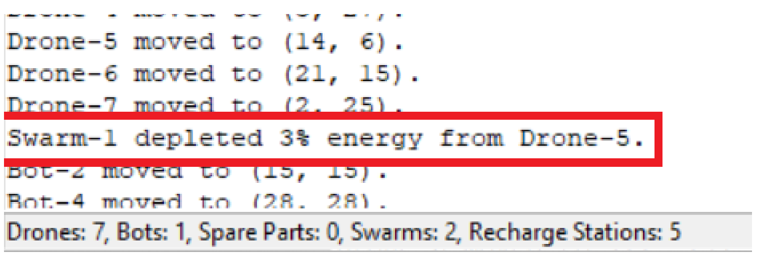
Figure 6: Drone energy depleting during pursue



```
Bot-1 moved to (7, 28).
Bot-5 moved to (23, 5).
Bot-6 delivered Part-24 to Station-2.
Bot-7 moved to (28, 3).
Drone-1 moved to (0, 12).
Drone-2 moved to (0, 9).
```

Drones: 7, Bots: 1, Spare Parts: 0, Swarms: 2, Recharge Stations: 5

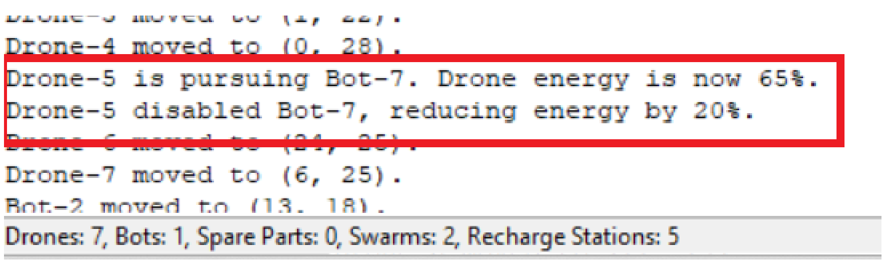
Figure 7: Bot performance



```
Drone-5 moved to (14, 6).
Drone-6 moved to (21, 15).
Drone-7 moved to (2, 25).
Swarm-1 depleted 3% energy from Drone-5.
Bot-2 moved to (15, 15).
Bot-4 moved to (28, 28).
```

Drones: 7, Bots: 1, Spare Parts: 0, Swarms: 2, Recharge Stations: 5

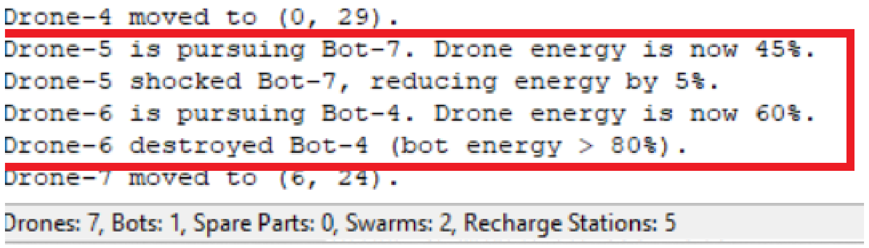
Figure 8: Swarm depleted energy of drone



```
Drone-4 moved to (0, 28).
Drone-5 is pursuing Bot-7. Drone energy is now 65%.
Drone-5 disabled Bot-7, reducing energy by 20%.
Drone-6 moved to (24, 25).
Drone-7 moved to (6, 25).
Bot-2 moved to (13, 18).
```

Drones: 7, Bots: 1, Spare Parts: 0, Swarms: 2, Recharge Stations: 5

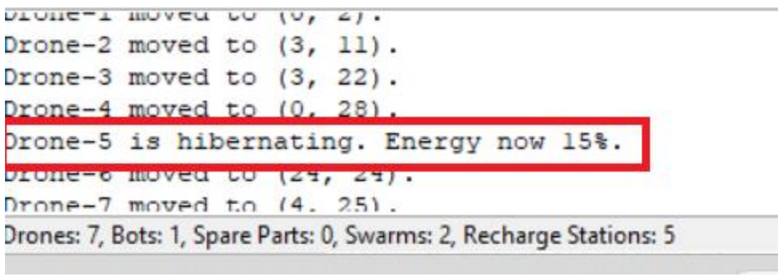
Figure 9: Drone performance according to energy level of bot



```
Drone-4 moved to (0, 29).
Drone-5 is pursuing Bot-7. Drone energy is now 45%.
Drone-5 shocked Bot-7, reducing energy by 5%.
Drone-6 is pursuing Bot-4. Drone energy is now 60%.
Drone-6 destroyed Bot-4 (bot energy > 80%).
Drone-7 moved to (6, 24).
```

Drones: 7, Bots: 1, Spare Parts: 0, Swarms: 2, Recharge Stations: 5

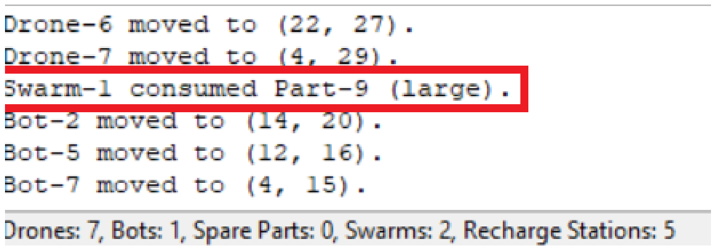
Figure 10: Drone performance according to energy level of bot



```
Drone-1 moved to (0, 2).
Drone-2 moved to (3, 11).
Drone-3 moved to (3, 22).
Drone-4 moved to (0, 28).
Drone-5 is hibernating. Energy now 15%.
Drone-6 moved to (24, 24).
Drone-7 moved to (4, 25).
```

Drones: 7, Bots: 1, Spare Parts: 0, Swarms: 2, Recharge Stations: 5

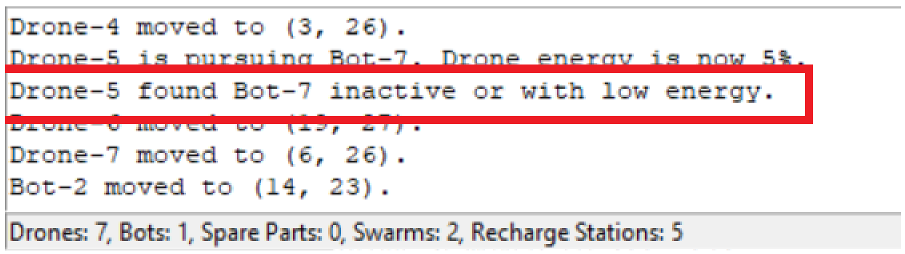
Figure 11: Drone hibernating to restore energy



```
Drone-6 moved to (22, 27).
Drone-7 moved to (4, 29).
Swarm-1 consumed Part-9 (large).
Bot-2 moved to (14, 20).
Bot-5 moved to (12, 16).
Bot-7 moved to (4, 15).
```

Drones: 7, Bots: 1, Spare Parts: 0, Swarms: 2, Recharge Stations: 5

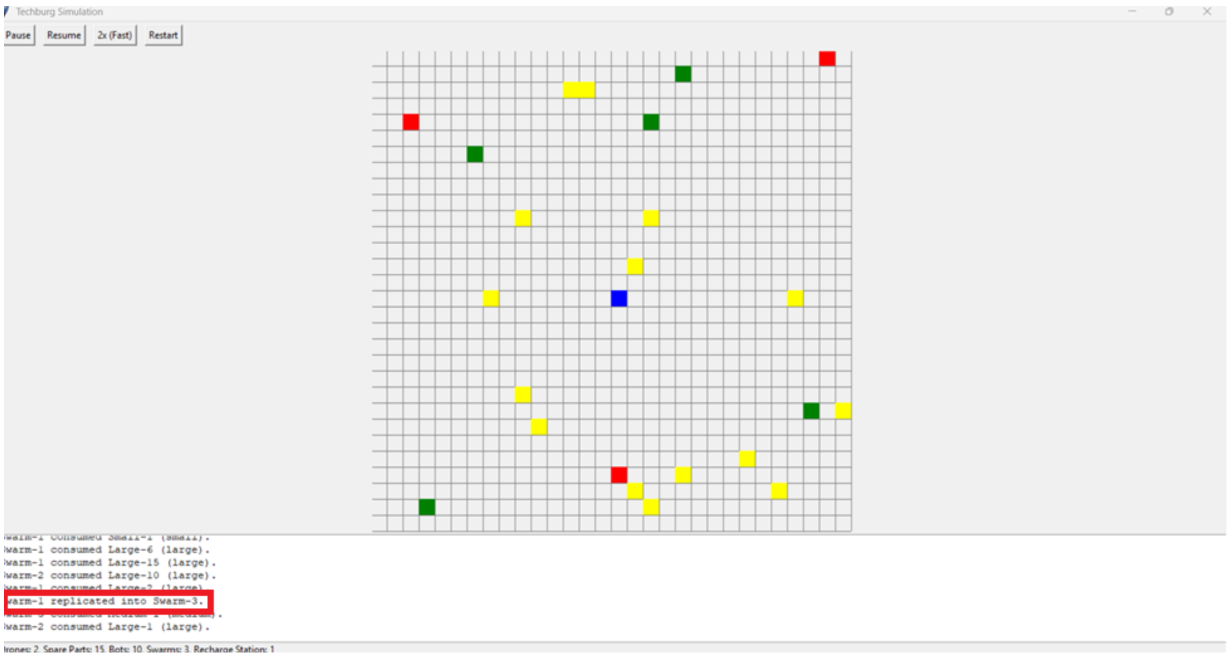
Figure 12: Swarm consumed corroded parts



```
Drone-4 moved to (3, 26).
Drone-5 is pursuing Bot-7. Drone energy is now 5%.
Drone-5 found Bot-7 inactive or with low energy.
Drone-6 moved to (19, 27).
Drone-7 moved to (6, 26).
Bot-2 moved to (14, 23).
```

Drones: 7, Bots: 1, Spare Parts: 0, Swarms: 2, Recharge Stations: 5

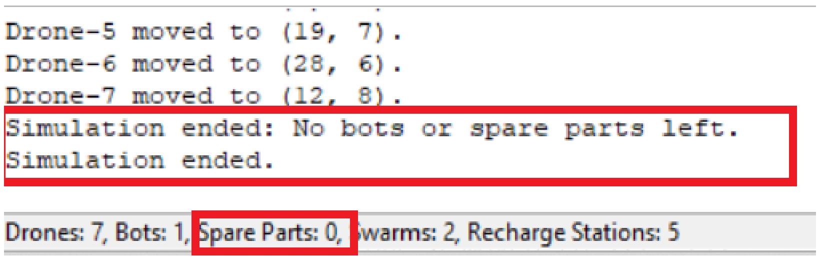
Figure 13: Inactive bot found due to low energy



```
warm-1 consumed Small-1 (small).
warm-1 consumed Large-6 (large).
warm-1 consumed Large-15 (large).
warm-2 consumed Large-10 (large).
warm-1 consumed Large-2 (large).
warm-1 replicated into Swarm-3.
warm-3 consumed Medium-1 (medium).
warm-2 consumed Large-1 (large).
```

rones: 2, Spare Parts: 15, Bots: 10, Swarms: 3, Recharge Station: 1

Figure 14: Swarm replication after consumption of parts

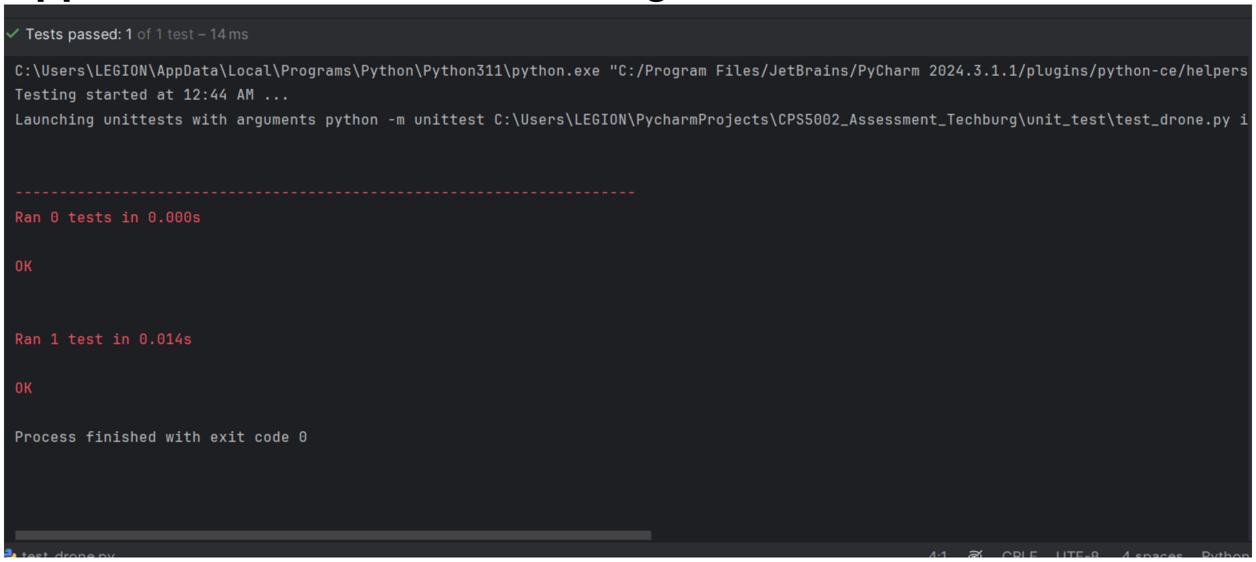


```
Drone-5 moved to (19, 7).
Drone-6 moved to (28, 6).
Drone-7 moved to (12, 8).
Simulation ended: No bots or spare parts left.
Simulation ended.
```

Drones: 7, Bots: 1, Spare Parts: 0, Swarms: 2, Recharge Stations: 5

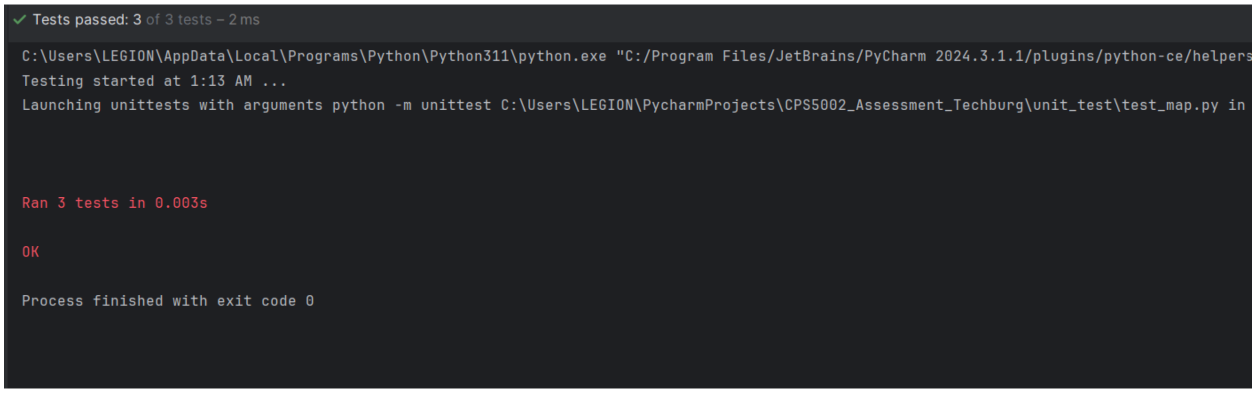
Figure 15: Simulation ended

Appendix C: Evidence of Testing



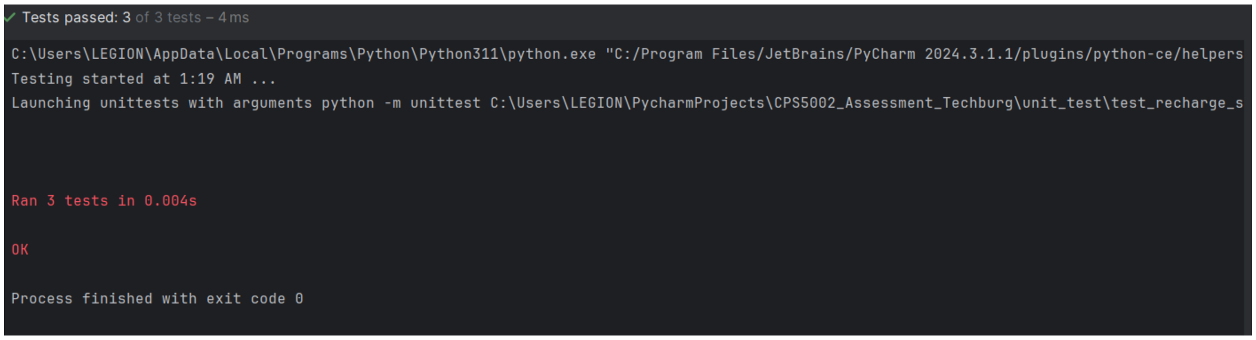
```
✓ Tests passed: 1 of 1 test – 14 ms

C:\Users\LEGION\AppData\Local\Programs\Python\Python311\python.exe "C:/Program Files/JetBrains/PyCharm 2024.3.1.1/plugins/python-ce/helpers
Testing started at 12:44 AM ...
Launching unittests with arguments python -m unittest C:\Users\LEGION\PycharmProjects\CPS5002_Assessment_Techburg\unit_test\test_drone.py i


----------------------------------------------------------------------
Ran 0 tests in 0.000s

OK


Ran 1 test in 0.014s

OK

Process finished with exit code 0
```

Figure 16: Unit testing drone class



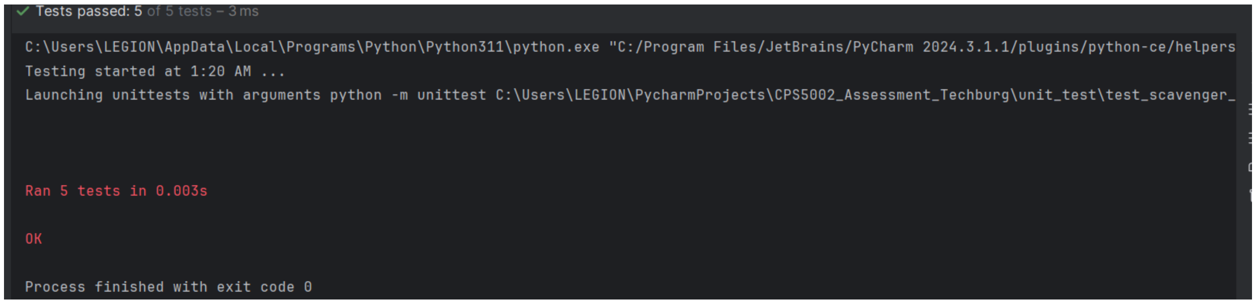
```
✓ Tests passed: 3 of 3 tests – 2 ms

C:\Users\LEGION\AppData\Local\Programs\Python\Python311\python.exe "C:/Program Files/JetBrains/PyCharm 2024.3.1.1/plugins/python-ce/helpers
Testing started at 1:13 AM ...
Launching unittests with arguments python -m unittest C:\Users\LEGION\PycharmProjects\CPS5002_Assessment_Techburg\unit_test\test_map.py in



Ran 3 tests in 0.003s

OK

Process finished with exit code 0
```

Figure 17: Unit testing map class



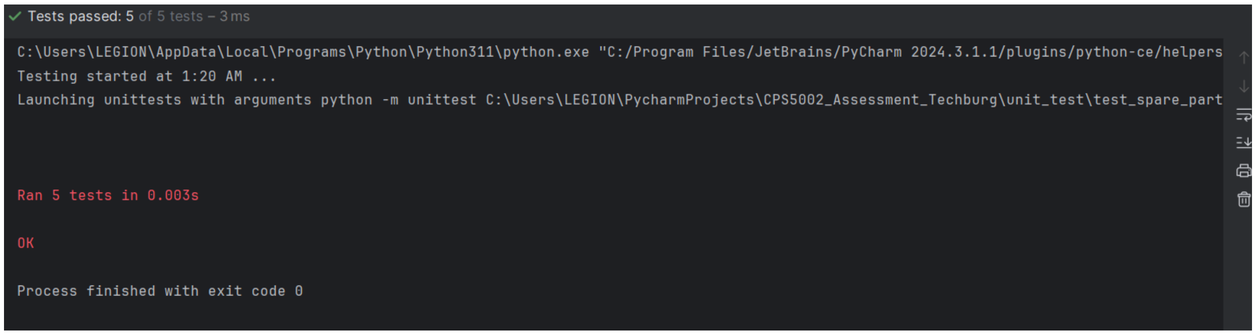
```
✓ Tests passed: 3 of 3 tests – 4 ms

C:\Users\LEGION\AppData\Local\Programs\Python\Python311\python.exe "C:/Program Files/JetBrains/PyCharm 2024.3.1.1/plugins/python-ce/helpers
Testing started at 1:19 AM ...
Launching unittests with arguments python -m unittest C:\Users\LEGION\PycharmProjects\CPS5002_Assessment_Techburg\unit_test\test_recharge_s



Ran 3 tests in 0.004s

OK

Process finished with exit code 0
```

Figure 18: Unit testing class recharge station



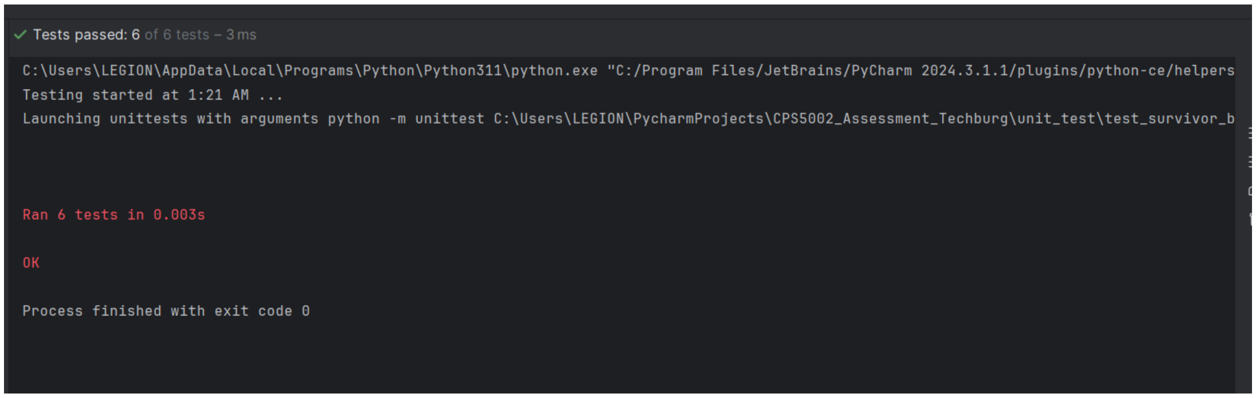
```
✓ Tests passed: 5 of 5 tests – 3 ms
C:\Users\LEGION\AppData\Local\Programs\Python\Python311\python.exe "C:/Program Files/JetBrains/PyCharm 2024.3.1.1/plugins/python-ce/helpers
Testing started at 1:20 AM ...
Launching unittests with arguments python -m unittest C:\Users\LEGION\PycharmProjects\CPS5002_Assessment_Techburg\unit_test\test_scavenger_



Ran 5 tests in 0.003s

OK

Process finished with exit code 0
```

Figure 19: Unit testing class scavenger swarms



```
✓ Tests passed: 5 of 5 tests – 3 ms
C:\Users\LEGION\AppData\Local\Programs\Python\Python311\python.exe "C:/Program Files/JetBrains/PyCharm 2024.3.1.1/plugins/python-ce/helpers
Testing started at 1:20 AM ...
Launching unittests with arguments python -m unittest C:\Users\LEGION\PycharmProjects\CPS5002_Assessment_Techburg\unit_test\test_spare_part



Ran 5 tests in 0.003s

OK

Process finished with exit code 0
```

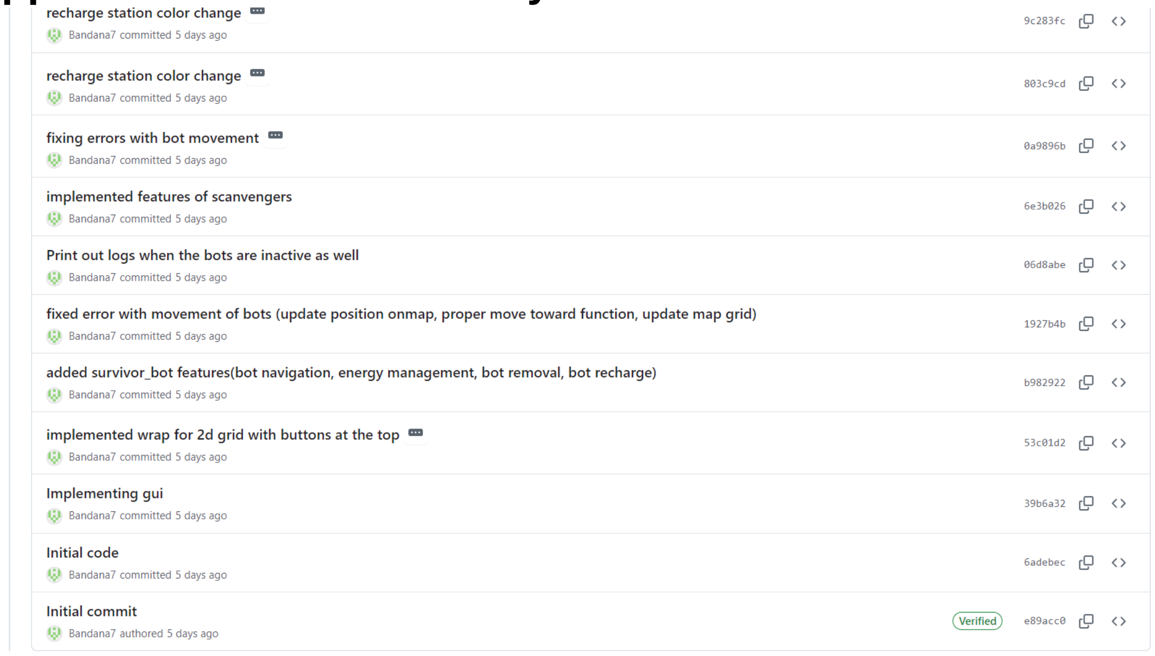
Figure 20: Unit testing class spare parts



```
✓ Tests passed: 6 of 6 tests – 3 ms
C:\Users\LEGION\AppData\Local\Programs\Python\Python311\python.exe "C:/Program Files/JetBrains/PyCharm 2024.3.1.1/plugins/python-ce/helpers
Testing started at 1:21 AM ...
Launching unittests with arguments python -m unittest C:\Users\LEGION\PycharmProjects\CPS5002_Assessment_Techburg\unit_test\test_survivor_b



Ran 6 tests in 0.003s

OK

Process finished with exit code 0
```

Figure 21: Unit testing class survivor bots

Appendix D: Commit History



recharge station color change
Bandana7 committed 5 days ago — 9c283fc

recharge station color change
Bandana7 committed 5 days ago — 803c9cd

fixing errors with bot movement
Bandana7 committed 5 days ago — 0a9896b

implemented features of scanvengers
Bandana7 committed 5 days ago — 6e3b026

Print out logs when the bots are inactive as well
Bandana7 committed 5 days ago — 06d8abe

fixed error with movement of bots (update position onmap, proper move toward function, update map grid)
Bandana7 committed 5 days ago — 1927b4b

added survivor_bot features(bot navigation, energy management, bot removal, bot recharge)
Bandana7 committed 5 days ago — b982922

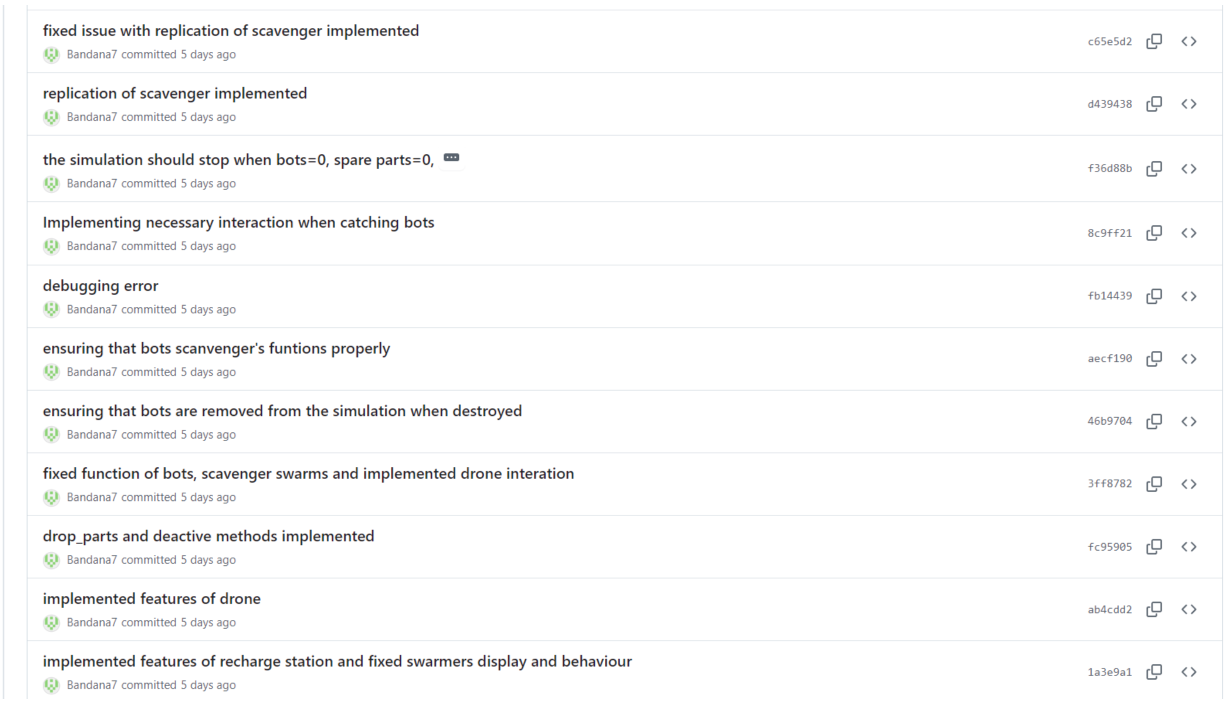
implemented wrap for 2d grid with buttons at the top
Bandana7 committed 5 days ago — 53c01d2

Implementing gui
Bandana7 committed 5 days ago — 39b6a32

Initial code
Bandana7 committed 5 days ago — 6adebec

Initial commit
Bandana7 authored 5 days ago — Verified — e89acc0

Figure 22: Commit history 1



fixed issue with replication of scavenger implemented
Bandana7 committed 5 days ago — c65e5d2

replication of scavenger implemented
Bandana7 committed 5 days ago — d439438

the simulation should stop when bots=0, spare parts=0,
Bandana7 committed 5 days ago — f36d88b

Implementing necessary interaction when catching bots
Bandana7 committed 5 days ago — 8c9ff21

debugging error
Bandana7 committed 5 days ago — fb14439

ensuring that bots scanvenger's funtions properly
Bandana7 committed 5 days ago — aecf190

ensuring that bots are removed from the simulation when destroyed
Bandana7 committed 5 days ago — 46b9704

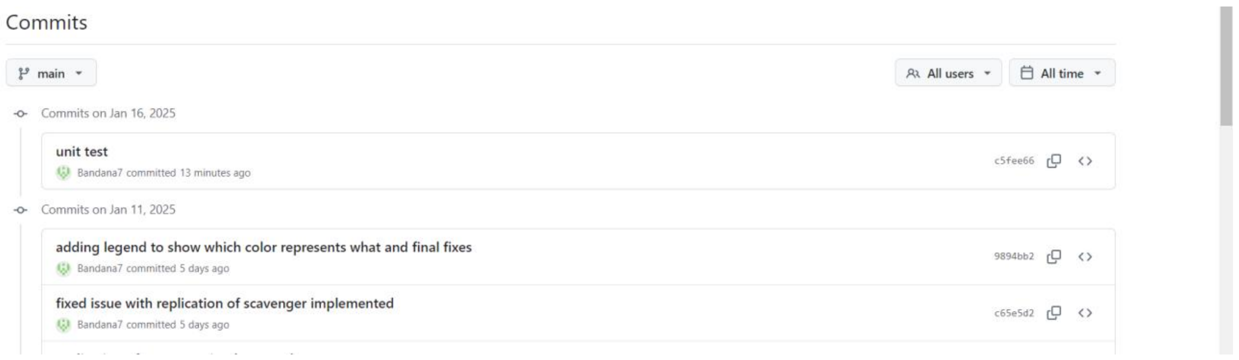
fixed function of bots, scavenger swarms and implemented drone interation
Bandana7 committed 5 days ago — 3ff8782

drop_parts and deactive methods implemented
Bandana7 committed 5 days ago — fc95905

implemented features of drone
Bandana7 committed 5 days ago — ab4cdd2

implemented features of recharge station and fixed swarmers display and behaviour
Bandana7 committed 5 days ago — 1a3e9a1

Figure 23: Commit history 2


Commits

main

All users

All time

Commits on Jan 16, 2025

unit test
Bandana7 committed 13 minutes ago
c5fee66

Commits on Jan 11, 2025

adding legend to show which color represents what and final fixes
Bandana7 committed 5 days ago
9894bb2

fixed issue with replication of scavenger implemented
Bandana7 committed 5 days ago
c65e5d2

Figure 24: Commit history 3