Research-centric Simulator for Swarm Robotic System

Planning Document

Jeremy Lim Shih Wen

19064732

BSc (Hons) in Computer Science

Department of Computing and Information Systems

Sunway University Kuala Lumpur

Supervisor: Dr Richard Wong Teck Ken

December 2, 2022

Abstract

Swarm robotic systems are inspired by the behaviours of natural swarms such as ant colonies and fish shoals. These behaviours are known as collective behaviours which allow these swarms to exhibit swarm properties including self-organization and decentralization. The replication of such swarm properties in swarm robotic systems is highly advantageous as it provides benefits like scalability, robustness, and flexibility to these systems. In nature, it is these swarm behaviours and properties that allow swarms to achieve tasks that no individual being from the swarm can achieve alone. Practically, swarm robotic systems are effective in various ranges of tasks which include tasks that cover large areas, are dangerous, and require scalability and redundancy. Although swarms in nature have evolved over millions of years, swarm robotic systems aim to accomplish the behaviours of swarms in as quick as possible. Therefore, the first step to achieve the replication of natural swarm in swarm robotic systems is the need for swarm robotic simulators to simulate and test algorithms on these systems. In this paper, a research-purpose swarm robotic simulator is proposed to fulfil the demand for swarm robotic simulators for research purposes, i.e., simulators that can conveniently test and debug algorithms, mainly for navigation tasks.

Keywords: Swarm robotics, simulator, research purpose, coordinated navigation.

Table of Contents

List of Figures

List of Tables

1 Introduction

The rise of robots integrated with artificial intelligence have been brought to attention in various industries as well as existing literatures, specifically robots that can determine routes, sense surroundings, generate local map data, and apply collected information to solve challenging tasks [1]. The swarm robotic system takes inspiration from the collective behaviour of social insects that organizes groups of robots with simple behaviours through a set of local rules [2]. These groups of simple robots can achieve tasks that no individual robot can succeed by themselves through the application of collective behaviours which allow for interaction among one another to accomplish common goals [3].

Although swarms in nature are the resultant outcome of millions of years of evolution, swarm robotic systems aim to achieve this in the shortest time possible. Therefore, the development of simulators for the simulations of these swarms are the beginning to the real implementation of swarm robotic systems for solving problems. Virtual simulations can be exploited for cost effectiveness because instead of using costly materials for robots to run experiments, the swarm robotic system can be tested in a simulation environment [4].

These complex systems would require thorough evaluations during simulations before its practical applications in real-life scenarios to avoid any propagation of errors that may lead to detrimental outcomes [5]. Additionally, researchers [6] acknowledge that majority of scholars make use of simulations and modelling for the validation and analysis of swarm robotic systems.

1.1 Aim

The aim of this project is to develop a simulation software for swarm robotic systems to test different swarm robotic algorithms for research purposes.

1.2 Objectives

1. To design/determine software architecture to allow integration/experimenting of different algorithms.
2. To implement a feature to ease the process of log and record results of the experiment on the simulation software.
3. To evaluate the simulation software in different applicable scenarios.

1.3 Project Scope

In scope:

- The simulation software supports Windows OS operating system.
- The simulation software will cover terrestrial swarms which require 2-dimensional planes.
- The simulation software will only take into consideration of swarm robotic systems.
- The simulation software will consider mainly for navigation tasks.

Out of scope:

- The simulation software supports other operating systems such as Android or iOS.
- The simulation software will cover aerial, aquatic, or outer space swarms which require 3-dimensional planes.
- The simulation software will implement the behaviours of real-world physics.
- The simulation software will consider for every swarm behaviour tasks.
- The simulation software supports integration of robot operating system (ROS) software.

2 Literature Review

2.1 Swarm Intelligence

2.1.1 Definition of Swarm

The term swarm is defined as individuals or beings that exhibit swarm behaviours which can include animals, insects, people, and so on. These swarms typically pertain to a single species; however, mixed swarms of different beings can also exist [7]. Another definition of swarm by researchers Barca and Sekercioglu [8] state that a swarm is a huge group of individuals that locally interact to accomplish common goals. Therefore, in the context of the field of robotics, a swarm would refer to swarm of robots that work similarly to natural swarms, i.e., locally interacting swarm of robots that possess common goals, which was first introduced by Beni [9] and Fukuda [10] in 1998.

2.1.2 Definition of Swarm Intelligence

In 1989, Beni and Wang [11-13] first devised the term swarm intelligence as a jargon in the field of cellular robotic systems to describe a group of independent and unsynchronised robots that coordinate actions to achieve a common goal. Currently, swarm intelligence has garnered more importance in recent years, especially in the communities of operations research and computational intelligence [14]. Swarm intelligence is defined by researchers Septfons et al. [15] as a group of simple agents with self-organized and decentralized interaction with each other to accomplish a common objective.

Since swarm intelligence does not have a specific definition, Dorigo and Birattari [16] denote swarm intelligence as a discipline that emphasises on collective behaviours of numerous individuals from artificial and natural systems which coordinate locally utilising decentralisation and self-organization within a specific environment.

Swarm intelligence is also defined as the ability for a whole swarm to learn or make decisions for a group as a single entity [13]. This inspiration comes from natural swarms that work together without direct communication between them such as ant colonies and fish shoals. For instance, ant colonies search for food by leaving pheromones to indicate to other ants about food sources whereas fish shoals migrate as a single entity from one place to another to protect each other from predators. Therefore, swarm intelligence refers to the intelligence of all the individuals in a swarm instead of one individual in the swarm [7].

Further, swarm intelligence brings about swarm algorithms that can simulate the natural swarm behaviours in hardware using real robots, software using computers, or metaheuristics for optimization problems in the artificial intelligence field [17]. Besides, swarm intelligence also generally refers to a collection of algorithms extensively applied in various industries and research to solve complex problems [14].

2.1.3 Swarm Algorithms

As mentioned in Section 2.1.2, swarm intelligence is concerned with swarm algorithms that are advantageous to various industries for problem solving. As noted by Khaldi and Cherif [2], the most popular swarm algorithms are Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Bee-based algorithms, and Artificial Fish Swarm Algorithm (AFS). The following algorithms are discussed in further detail.

The development of PSO algorithms were inspired by the behaviours from birds and fishes that travel long distances together in flocks or schools to seek for food sources [2]. PSO algorithms are also commonly applied for analysis and modelling purposes [18-20].

The ACO algorithm is inspired from the behaviours of ant colonies foraging by determining the most optimal path to travel from the nest to the food sources [2].

AFS algorithm is inspired by the collective movement behaviours observed in fishes when they search for food, perform stochastic searching, and protect each other against predators [21].

The Bee-based algorithms are inspired from the behaviours of bees of three sections, firstly on the mating behaviour of honeybees, secondly on the foraging behaviour of honeybees, and thirdly on the evolution process of queen bees [22].

These swarm algorithms are practical for various purposes such as for the predictions of swarm behaviour, coordinating robots, researching robots, and the modelling of swarm robotics [7].

2.2 Swarm Robotics

2.2.1 Definition of Swarm Robotics

The terms swarm robotics and swarm robotic systems are noted to be used interchangeably. Primarily, the term swarm robotics was described by Beni [12] as the coordination of a group of robots with unique characteristics such that it follows the behaviours of insect swarms, i.e., relatively simple, identical, unsynchronized, and decentralized. The main inspiration of swarm robotics is derived from the behaviours of social insects such as ants, termites, and wasps which coordinate themselves to achieve tasks that are impossible alone [23, 24]. Examples include ants working together to carry large objects whereas wasps cooperating to build huge mounds from mud while maintaining moisture and ideal temperatures [25].

Generally, according to [12, 24, 26], swarm robotics is defined as a system satisfied by the following conditions, i.e., i) the number of robots in a swarm robotic system must be immense, ii) the tasks completed by the swarm of robots are beyond the capabilities of an individual robot, iii) to accomplish tasks, robots utilise local information to communicate with each other within short distances.

Swarm robotics is also closely related to term swarm intelligence such that both terms are concerned with decentralized and self-organized systems [2]. However, the main difference between swarm robotics and swarm intelligence is that swarm robotics emphasises on the physical application of real robots in a realistic environment [27].

Therefore, researchers Osaba et al. [14] suggest that swarm robotics is the application of swarm intelligence on a population of physical robot devices under various scenarios to complete different tasks that are incapable of completion by any individual robot. Likewise, according to [28], swarm robotics utilises the methods of swarm intelligence to solve real word problems using real mobile robots.

2.2.2 Features/Properties of Swarm Robotic Systems

The two main features in swarm robotic systems are decentralization and self-organization [28]. Decentralization implies that robots only interact among themselves and do not use global information from the main controller or dominion to complete assigned tasks in the environment [29, 30]. Decentralization basically describes that there is no leader in the swarm of robots to command other robots to perform tasks, and that the robots communicate

locally to accomplish the task. Additionally, the decentralized feature of a swarm means that the loss of any robot in the swarm will not affect the operation of the swarm [23].

Self-organization is a dynamic technique according to the quantity of robots to maintain a designed structure of the system [31]. Self-organization operates on four rules which are randomness, interactions, positive feedback, and negative feedback [32]. An example of self-organization can be seen in the ant foraging behaviour, whereby ants travel on random paths to forage food, and deposits pheromones on paths they have travelled, using the path with the most deposited pheromones as the most optimal path [29]. In this case, randomness is determined by the random paths that the ants chose, interactions are the pheromones deposited by ants that have travelled through paths, positive and negative feedback depends ants' ability to choose paths based on the concentration of pheromones left by ants, with higher concentration as more optimal path and lower concentration as less optimal path to food sources.

2.2.3 Advantages of Swarm Robotic Systems

The main motivation behind the idea of swarm robotics are the three significant and highly demanded advantages that are deemed to provide by the system, i.e., scalability, robustness, and reliability [33]. Thus, Beni [12] proposed that a swarm robotic system must display properties of scalability, robustness, and flexibility, to efficiently replicate the idea of natural swarming.

Primarily, scalability indicates that a system should work as intended when the number of groups increases. It is crucial that the number of robots in a swarm should not be fixed and that the swarm should still be able to accomplish tasks effectively as the number of robots differs. The number of robots in a swarm should not be detrimental to the performance of the system. Therefore, a swarm robotic system works effectively with different number of robots in the system regardless of the size of swarm [12].

Furthermore, robustness refers to the ability of a system to continue operating without fail despite any occurrence of system faults or environmental disturbances. System faults occurs when one or more robots in a swarm of a system that has malfunctioned or unable to act whereas environmental disturbances occur due to changes in the surroundings such as obstacles, weather patterns, and others. It is vital for swarm robotic systems to adapt to such conditions that may occur in an environment. The general performance of a swarm robotic

system should not be affected in the event where few individual robots of the swarm have malfunctioned such that other robots present in the swarm can compensate for the malfunctioned robots and continue to maintain the efficiency of the system [12].

Finally, flexibility implies that a system is flexible by being able to adapt to changes in an environment. Individual robots in a swarm robotic system must cooperate and coordinate among each other to find solutions when dealing with tasks. To elaborate, when proposed with tasks, robot swarms should adapt by changing roles depending on the assigned tasks and work collectively to find an appropriate solution to the task. Therefore, a swarm robotic system should exhibit flexibility by being able to act concurrently according to any changes that occur in the environment [12].

2.2.4 Collective Behaviour Tasks

The collective behaviours of a swarm robotic system can be categorized into four types, i.e., navigation behaviours, spatially organizing behaviours, collective decision making, and other collective behaviours [26]. Hence, the three primary swarm robotic behaviours are discussed in this section.

2.2.4.1 Navigation Behaviours

Navigation behaviours are concerned with how the robots are moving or travelling together as a group [34]. This section covers three navigation behaviours which are collective exploration, coordinated motion, and collective transport.

In collective exploration, according to [34], the application of robots to explore an unknown area is a fundamental problem in the navigation of robots in the robotics field. However, for swarm robotic systems, it is required for the deployment of not only one but several swarm robots to explore an undiscovered area by utilising cooperation and navigation in the swarm. Additionally, exploration of new areas can be achieved more efficiently through the deployment of more than one robot with an additional benefit of fault tolerance to guarantee the completion of the task. Further, the capabilities of swarm robots to share information locally can handle uncertainties and sensor biases during the exploration. Researchers also noted that collective exploration is effective at achieving collective mapping and path optimization of large spaces. As such, collective exploration is defined as the ability of a robot

swarm to be able to cooperate, explore, and achieve tasks such as space exploration, surveillance, and so on, in an environment [26].

In coordinated motion, according to [35], coordinated motion refers to the preservation of a formation in a robotic swarm during motion which is also known as flocking. The preservation of formations of a moving robotic swarm is effective for preventing collisions between one another and obstacles in the environment which acts as a navigation mechanism for the swarm robotic system during exploration. This mechanism is referred to as obstacle avoidance which aims to prevent any collisions in the swarm robotic system. Path planning is employed for the robots to navigate and avoid obstacles in the environment.

The collective transport behaviour is inspired by the capability of ants to transport objects that are multiples times heavier than an individual ant from an initial location to a goal location. In swarm robotic systems, this task can be achieved by employing probabilistic finite state machines and artificial evolution techniques. The robots either communicate explicitly or implicitly to calculate the amount of force exerted by neighbouring swarm robots to move the object [34]. Additionally, through artificial evolution, collective transport can be achieved by the manipulation of robot control laws [36, 37].

2.2.4.2 Spatially Organizing Behaviours

Spatially organizing behaviours explain how robots can organize and disperse in an area using unique methods such as chain formation, pattern formation, aggregation, object clustering and assembling, and self-assembly and morphogenesis [34]. This section covers the various techniques of spatial organization of robots in swarm robotic systems.

Chain formation refers to the formation of robots in a chain structure. The benefits of chain formation are that is allows for more efficient exploration of new territories and eliminating a single point of failure within the swarm robotic system. Some scenarios that require the application of chain formation of swarm robots are the surveillance of narrow areas such as tunnels, sewers, caves, and so on [38].

The behaviour which allows robots to coordinate to form patterns and shapes such as rectangles, circles, and more, is known as pattern formation. Robots are deployed in a specific manner and distance within one another to achieve unique formation for the specific pattern [7].

Aggregation, one of the most observed and studied swarm behaviours of nature, refers to the task where every robot positions themselves closely in a specific location [39]. There are two types of aggregation which are cue-based and self-organized [29]. Cue-based aggregation refers to accumulating cues based on the external environment which include cues such as light and temperature [40]. Self-organized aggregation refers to capability of a swarm to be able to organize and gather in a random location without using cues [41]. It is noted that a combination of both types of aggregation are necessary to increase the effectiveness of swarm robotic systems in practical applications [42]. Furthermore, swarm robotic systems require aggregation behaviours because swarm robots must be in proximity of each other in most cases [43]. One approach to achieve aggregation in swarm robotics is through the application of evolutionary robotics such as using genetic algorithms to train neural networks for a swarm of robots on an aggregation task [41, 44].

Object clustering and assembling refers to the ability of robots to be able to gather randomly spread objects from an environment and assemble the objects in a specific area. This behaviour is applied in constructive construction to create two-dimensional or three-dimensional arrangements [45-47]. An example of object clustering and assembling is the efficient storage of goods in a warehouse by swarm robots.

Self-assembly describes the potential for robots to physically attach to each other and create a specific formation. This behaviour is highly advantageous to swarm robots by increasing the pulling strength of robots on objects, improving stability of robots on uneven terrains, creating a structure to guide other robots, and being able to form structures to overcome gaps in spaces to prevent falling [35].

2.2.4.3 Collective Decision-Making

Collective decision making determines how a swarm of robots can make choices as a group by building a consensus and allocating tasks according to the problem faced [34]. This section discusses on two collective decision-making techniques employed by swarm robotic systems, i.e., task allocation and consensus achievement.

Task allocation is a decision process for allocating different tasks to each robot in the swarm according to a list of alternative tasks to be performed. An example of task allocation in the foraging task is presented by [48] where robots are assigned to collect food particles from the environment and bring them back to specific location. The efficient task allocation in

this situation is for the robots to break down the task into two subtasks, i.e., transport food from the source to a main storage location, and transport food from the main storage location to the nest. A dynamic task allocation technique was introduced to distribute the tasks among the robots such that one part of the swarm performs the first task, and the remaining of the swarm perform the other task to increase transport efficiency.

Consensus achievement refers to how robots generally agree to a decision such as on path or target selection. Robots can either use explicit communication through the exchange of messages or implicit communication through local information to generate the consensus. This behaviour is useful in situations where a consensus is needed by the swarm of robots to effectively accomplish the task [35].

2.2.4.4 Overview of Swarm Robotic Behaviours

An overview of swarm robotic behaviours is shown in Table 1. Each behaviour is classified into their respect categories and a short summary is provided to assist in the understanding of the behaviours.

Table 1 Overview of swarm robotic behaviours.

Behaviour Classification	**Swarm Behaviour**	**Summary**
Navigation behaviours	Collective exploration	Robots efficiently cooperate, explore, and achieve tasks in an environment.
	Collective motion	Robots preserve formation during motion and avoid obstacles with path planning.
	Collective transport	Robots transport objects that are heavier than an individual robot using explicit or implicit communication between neighbouring robots to exert force.
Spatially organizing behaviours	Chain formation	Robots form a chain formation to efficiently explore narrow spaces.
	Pattern formation	Robots coordinate to form patterns by being deployed in specific manner and distance between each robot.

	Aggregation	Robots position themselves closely in a specific region using cues from the environment or by self-organization.
	Object clustering and assembling	Robots collect randomly spread objects in an environment and assemble them in a particular location.
	Self-assembly	Robots attach to one another to increase their pulling strength, improve stability, create structures for other robots and overcome holes in terrains.
Collective decision-making	Task allocation	Robots distribute and allocate tasks efficiently among each robot to achieve a task.
	Consensus achievement	Robots generally agree to a certain decision from various alternatives using the available explicit or local information.

2.2.5 Applications of Swarm Robotics

2.2.5.1 Tasks that cover a large area

Swarm robotic systems are effective at tasks that require robots to explore large areas or spaces. Examples of these tasks include search and rescue, demining, surveillance, and so on. Typically, swarm robots are dispersed in large areas such as extra-terrestrial or underwater areas with limited information on the area's infrastructure to guide the robots to perform these tasks. Swarm robotic systems excels in exploration tasks because each robot possess the ability to move autonomously without information of the infrastructure, identify and examine dynamic alterations of the environment, pinpoint sources, move towards the pinpointed sources, and perform brief actions [2].

2.2.5.2 Tasks that are risky for robots

In risky tasks such as recovery missions, robots may be lost or irretrievable after the accomplishment of tasks. Swarm robotic systems are useful at these tasks because these systems comprise of cheap and simple individual robots which are more economically efficient compared to using expensive and complex robots for these tasks [2]. Therefore, swarm robotic systems are effective at dangerous tasks because they will not cause a huge expense financially due to the cost efficiency of each individual robot.

2.2.5.3 Tasks that involve scalable population size and redundancy

Another area that swarm robotic systems excel at are in scenarios that are challenging to approximate the number of resources required to complete the task such as cleaning or search and rescue. For instance, in an oil spill scenario, the number of resources required varies frequently with high spillage at the start, and lower spillage at the end. The advantage of swarm robotic systems in this scenario is the scalability and flexibility to add or remove robots for cleaning the spillage without degrading the performance of the system and accomplishing the task. Hence, the removal of redundant robots when requirements of the task decreases is highly beneficial as it saves resources.

2.3 Research-centric Swarm Robotic Simulator

2.3.1 Purpose of Simulator

Before a swarm robotic system is deployed for real-life applications, tests are performed to ensure that the system works as intended to accomplish the assigned tasks, however, the testing of such systems are extremely costly. Therefore, the main purpose of developing swarm robotic simulators is to save cost and time on testing with analysis and modelling of swarm robotic systems [7]. Another benefit of simulators is that when satisfactory output is obtained, the model can be conveniently transferred to the physical robot swarm [7]. In contrast, if a model does not produce desirable results, a simulator can utilise controlled repetition to repeat the experiment such that the model can be inspected in detail to understand the underlying problems in its design [49]. Therefore, researchers found that majority of the early swarm algorithms were tested using simulators instead of physical robots [7].

Despite the abundance of benefits of swarm robotic systems, researchers discovered that it is often challenging to achieve a behavioural design of a robot that attains a satisfactory global performance for the system [50]. Therefore, Calderón-Arce et al. [51] emphasised on the importance of existence of simulators as the intermediary between simulation and reality because simulators allow for simulation, learning, and modelling of such physical swarm robotic systems.

Simulators are important for the optimization of swarm robotic systems in terms of capabilities and algorithms by analysing their performance through virtual experiments and swarms, and as a result, the optimal design of a swarm can be determined without cost and time-consuming tests on real robots [50]. Researchers established that simulations are highly advantageous for testing algorithms on a large scale due to the low accessibility of constructing a substantial number of real robots [2].

Although simulators allow users to perform experiments more conveniently, safer, and quicker than experiments on physical robots, researchers mentioned that it is challenging to determine a suitable software for the simulation of swarm robotic systems [51]. Researchers Erez et. al. [52] emphasised that simulators performed most effectively according to the respective objective of the simulator that it was developed and optimized for.

2.3.2 Simulator Limitations

Researchers generally agree that in swarm robotic systems, simulators are unable to account for all aspects of reality into the simulation [17, 51]. According to [51], one of the reasons is that the actuators and sensors of virtual robots are noise and interference free. Another reason is that the simulation does not consider for unpredictable environmental factors such as temperature, wind, friction, dust, and luminance. Furthermore, physical characteristics of the robots such as battery life, degradation of robot parts, and energy level are not replicable in simulations.

For these reasons, researchers stated that a successfully tested swarm robotic system in simulators will not certainly be successful when implemented in real life [17]. Therefore, simulations of robots and environments are still incapable of accurately replicating reality due to unpredictable factors and details which can occur in the real world [51].

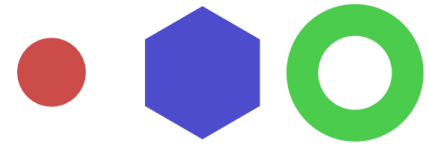
2.3.3 Evaluation of Existing Simulators

In this section, similar swarm robotic simulators are evaluated according to their objectives, tasks covered, user interface, availability, and visualizations.

2.3.3.1 Swarm-Sim

Swarm-Sim [53] is a round-based simulator that was developed for algorithm testing on robot swarms for the research field. It allows for the simulation of 2D and 3D large scale swarms as well as testing and evaluation of the algorithms on the swarms using plots and visualizations. *Swarm-Sim* is supported on three operating systems which include *Linux*, *MacOs*, and *Windows*. *Swarm-Sim* is also open-source and fully written in the Python language which brings convenience in implementing swarm algorithms for simulations. Regarding the models for matter, *Swarm-Sim* represents their agents as red circles, items as blue hexagons, and location as a green and white circle as seen in Figure 1. The simulator uses *OpenGL API* for their visualizations and *PyQt5* library for the graphical user interface as shown in Figure 2. *Swarm-Sim* supports many use cases because it designs their agents such that they can navigate a simulated world, observe their surroundings, pick up items and agents, and interact with cues, items, and other agents in the world. Examples of use cases of *Swarm-Sim* include the coating of randomly shaped objects, the flocking of swarm robots, the demonstration of lawn mower

robots, and the simulation of phototactic movement of swarm robots. Figure 3 illustrates the simulation use cases of coating of randomly shaped objects and swarm flocking.



(a) Agent (b) item (c) Location

Figure 1 Model visualization in Swarm-Sim.

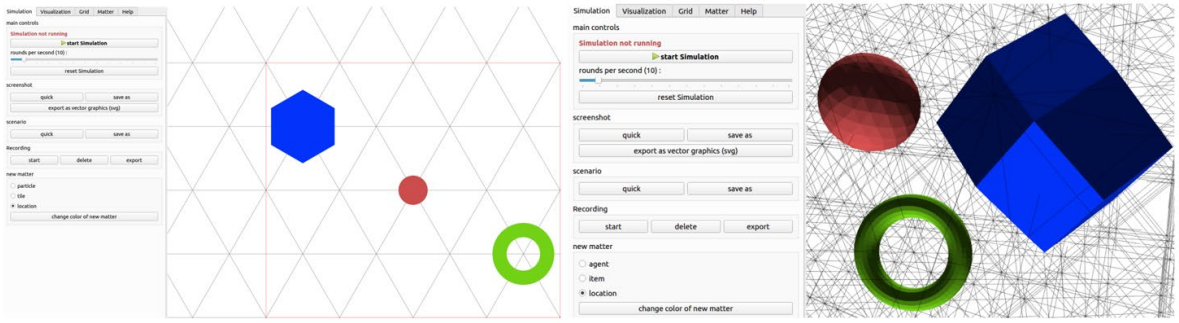


Figure 2 2D (left) and 3D (right) of Swarm-Sim graphical user interface.

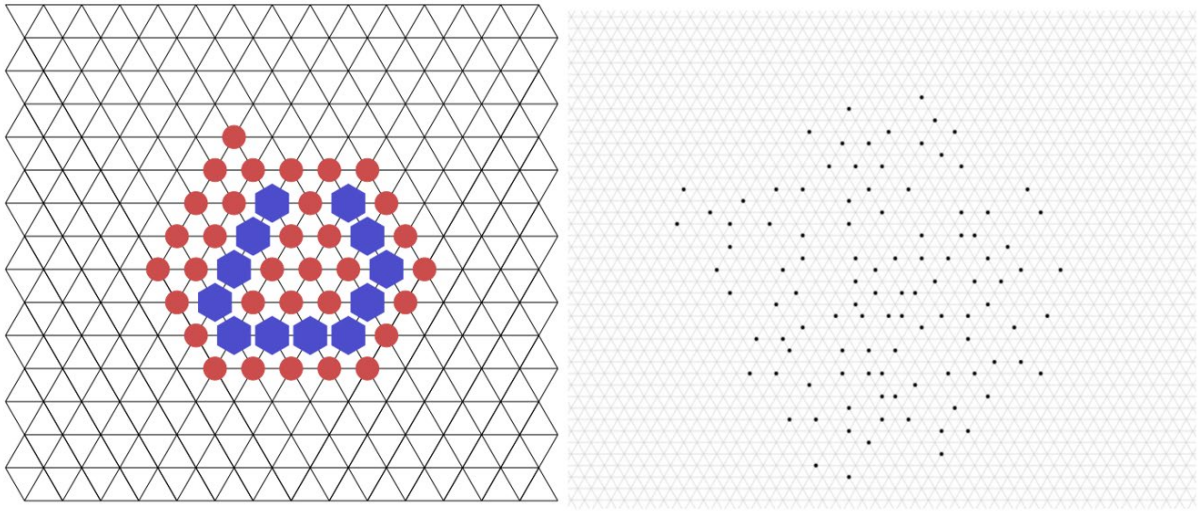


Figure 3 Coating of randomly shaped objects (left) and Swarm flocking (right).

Regarding the debugging and evaluation aspect for Swarm-Sim, the speed of the simulation can be controlled, and the memory of matter can be inspected and interacted with which allows for the analysis of the model and algorithm. Once a simulation is completed, Swarm-Sim produces five outputs for evaluation and reproduction of scenarios. Firstly, three types of CSV files are generated. The first file stores each round of data from the simulation, the second file stores the actions of each agent during each round, and the third file contains a summary of aggregation which represent different seeds with the same solution, information such as standard deviation and average round are included. Next, Swarm-Sim creates plots of PNG images from the simulation and stores it in the plot folder. Further, users can take screenshots while the simulator is running, and screenshots are stored in the screenshots folder. Besides that, users can take videos of the simulations which are stored as avi or mp4 files. Lastly, the log file which includes Swarm-Sim's internal calls are saved in the log folder. Hence, generated outputs from the simulation enables users to comprehensively evaluate the model and algorithm of the swarm robotic system.

2.3.3.2 SwarmLab

SwarmLab [54] is an open-source platform fully written in *MATLAB* for drone swarm simulations, also known as aerial swarm simulations. The authors of the paper stated that *MATLAB* was selected because the language provides high abstraction and convenient built-in functionalities for the development of the system. The aim of abstraction in the scripting language is to encourage scientific communities to make use of the simulator for research purposes. The main purpose of *SwarmLab* is to solve the lack of simulation frameworks for debugging, analysis, optimizing, testing of algorithms. The simulator was designed such that users can modify the parameters of the simulation, drone, swarm, and map, as well as edit codes and test the program within one scripted programming language. Furthermore, the simulator was designed for collective navigation tasks and allows side-by-side comparison of two algorithms with performance analysis of both algorithms. *SwarmLab* also provides a feature to translate embedded codes to *C* or *C++* to improve computational performance of the simulator. The graphical user interface of the simulator is shown in Figure 4 where users can edit the parameters of the simulator. The 2D visualization of the simulator are also shown in Figure 5 and the 3D visualization shown in Figure 6.

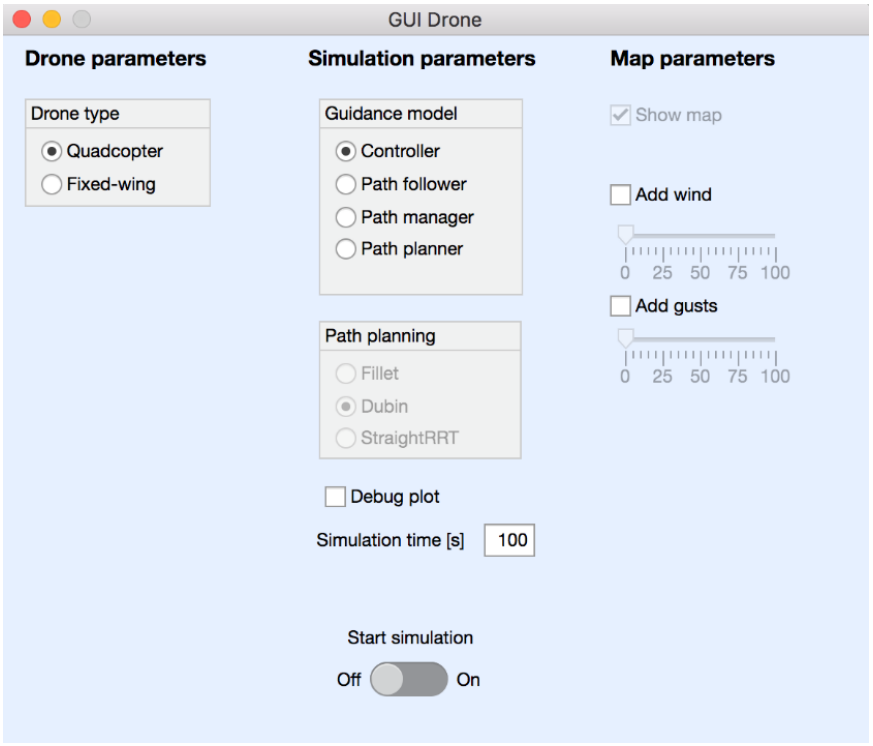


Figure 4 Graphical user interface of SwarmLab.

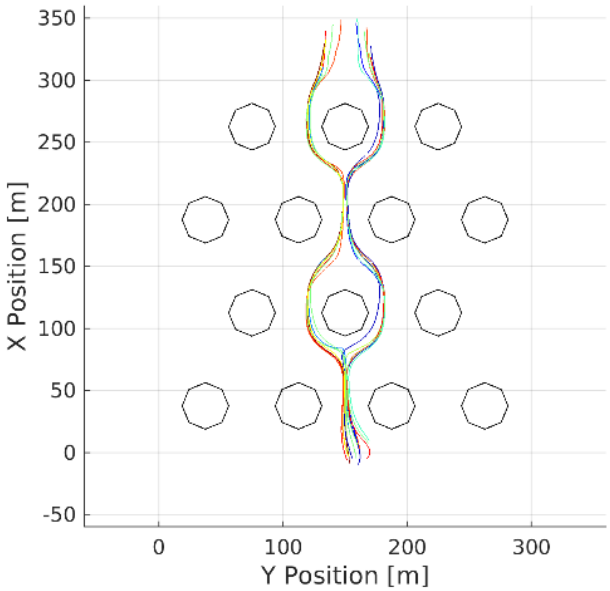


Figure 5 2D visualization of SwarmLab.

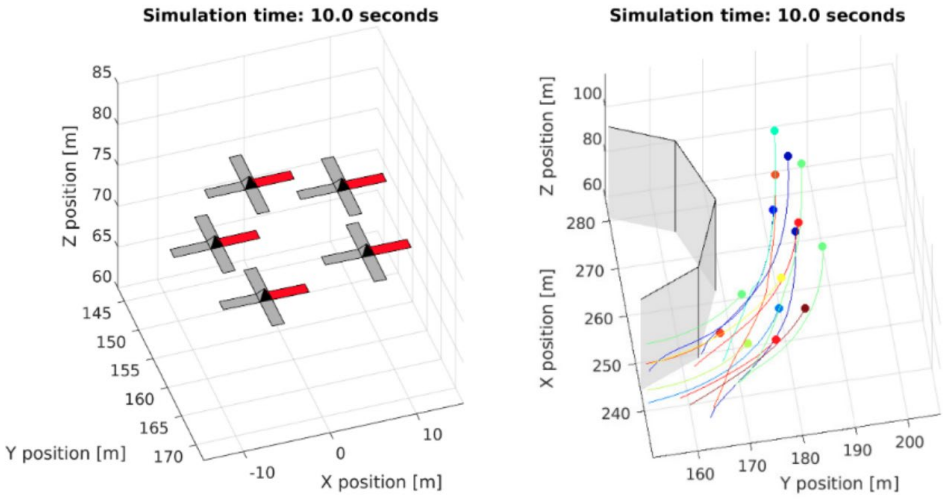


Figure 6 3D visualization of SwarmLab.

2.3.3.3 BeeGround

According to [55], *BeeGround* is an open-source swarm robotic simulator built on *Unity Development Engine* for swarm robotics research. The aim of *BeeGround* is to improve accessibility of swarm robotic simulators and to develop a simulator to run swarm simulations conveniently and efficiently. *BeeGround* offers a variety of experimental parameters to be modified by the user. These parameters include population size, robotic platforms, arena size, swarm design, and conditions of the environment such as obstacles, humidity, and temperature, as well as swarm behaviour. It also supports platforms for ROS projects through the integration of *ROS-Unity* packages. Furthermore, *BeeGround* supports the interaction of agents with the environment, thus, tasks such as pheromone deposition of ant colonies can be performed. For instance, in [55], the honeybees' aggregation was simulated to showcase and test simulator's capabilities. The model used by the simulator is an abstract model of a real-life robot, Mona as shown in Figure 7. The simulator's user interface is shown in Figure 8 where users can modify parameters, examine simulations, edit object hierarchy, import assets, and view debugging console. The sample simulation is illustrated in Figure 9 which showcases the swarm aggregation behaviours at different timing.

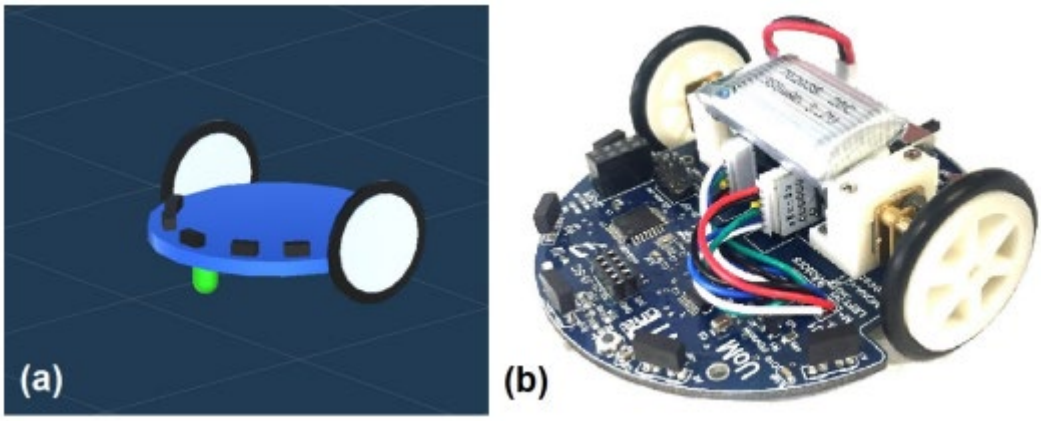


Figure 7 Mona, an abstract model for BeeGround.

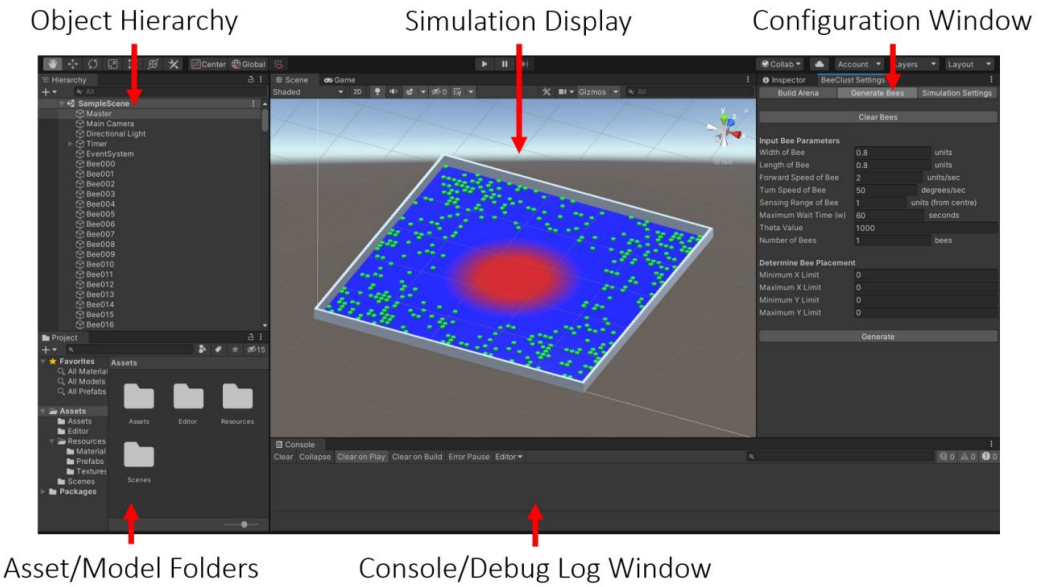


Figure 8 Graphical user interface of BeeGround.

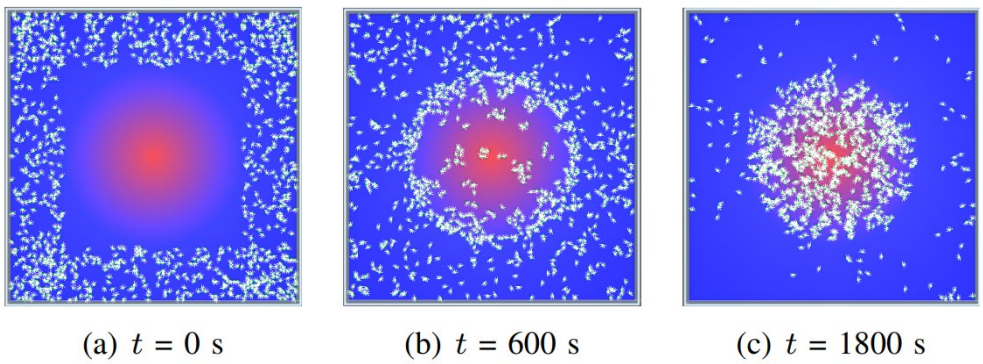


Figure 9 2D visualization of BeeGround.

2.3.4 Comparison of Existing Simulators

This section includes a table for the comparison of existing research purpose swarm robotic system simulators. The features of the simulator such as availability, objectives, tasks covered, user interface, and visualizations are compared as shown in Table 2.

Table 2 Comparison of existing research-purpose swarm robotic simulators.

Simulator	**Availability**	**Objective(s)**	**Task(s) covered**	**User interface**	**Visualization of simulation**	**Reference**
Swarm-Sim	Open-source	Algorithm testing for research purposes.	Spatially organizing behaviours, others	Simple, easy to understand	Abstract in 2D / 3D	[53]
SwarmLab	Open-source	Provide abstract simulator to encourage scientific communities to use it for research purposes and to solve the lack of frameworks for evaluating swarm algorithms.	Navigation behaviours	Simple, easy to understand	Abstract in 2D	[54]
BeeGround	Open-source	Improve accessibility of swarm simulations for swarm robotics research and to develop a convenient and efficient simulator.	Spatially organizing behaviours, others	Sophisticated, requires additional knowledge about swarm simulation	Realistic in 2D / 3D	[55]

2.4 Important Findings

The important findings of the literature review include several guidelines for the swarm robotics simulator.

Firstly, the differences between swarm intelligence and swarm robotics were determined such that swarm intelligence refers to ability for a swarm to make decision whereas swarm robotics is more towards the description of physical robots such that the robots utilise swarm intelligence to achieve common goals.

Furthermore, the collective behaviours of swarms were discussed in detail such that it can be divided into three sections, i.e., navigation, spatially organizing, and collective decision-making behaviours. Each of these sections describe how the behaviours of natural swarms can be applied onto robotic swarms. It is also noted that these behaviours are what makes a swarm robotic system different than multi-robot systems.

Finally, several existing research purposes swarm robotic simulators were evaluated. Features such as their objectives, availability, tasks covered, user interface, and visualizations were closely examined. The important features of each simulator are then accumulated and proposed in the technical plan for the development of a swarm robotic simulator.

3 Technical Plan

3.1 Software Development Life Cycle

A software development life cycle is introduced to ensure that the project will be successfully executed. The software development life cycle model adopted in the capstone project is the waterfall model. The waterfall model is advantageous because the project can be completed in phases which accommodates to the two-part gap in the capstone project. The phases of the waterfall model begin with requirements, design, implementation, testing, deployment, and maintenance, in that respective order. Each phase of the waterfall model has deliverables which need to be completed before moving on to the next phase. The waterfall model phases are split into two parts, one part consists of requirements and design for Capstone Project 1, and the remaining part consists of implementation, testing, and deployment for Capstone Project 2 as shown in Figure 10. It is noted that the maintenance phase is not necessary because the capstone project will end before requiring any maintenance. The duration of the project is 28 weeks whereby both Capstone Project 1 and Capstone Project 2 are allocated 14 weeks each.

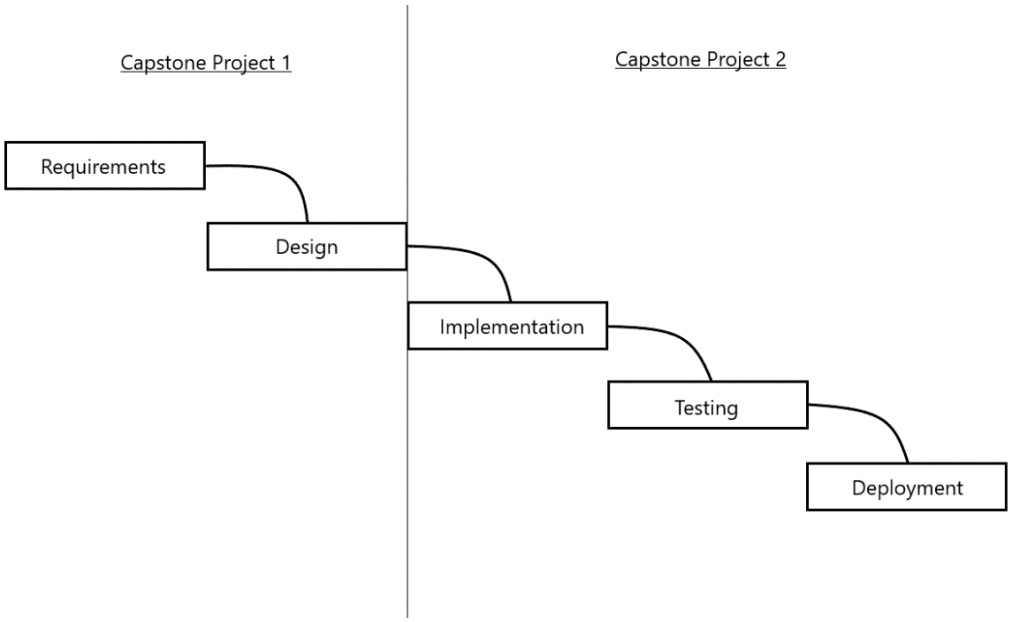


Figure 10 Software development life cycle for capstone project.

In the requirements phase, detailed software functionalities of the system are gathered and analysed for application on the system. The requirements consist of functional and non-functional requirements which determines the capabilities and nature of the system. In this case, the requirements are gathered based off evaluations of similar simulators from existing literatures.

The design phase is where requirements are transformed into a viable solution for the system. This phase covers the project objectives, schedule, wireframe of the system, designs of algorithms and models, and unified modelling language diagrams of the system.

The implementation phase is where the project plan is executed to achieve the desired proposed system. For this project, the proposed system that will be implemented will be a swarm robotic simulator for research purposes.

Regarding the testing phase, the developed system will undergo various test cases to ensure that the system satisfies the proposed requirements. Test reports are generated in this phase to assess the quality of the developed system.

The final phase is the deployment phase where the completed project is deployed to users. In this case, a research-centric simulator for swarm robotic system is deployed to the customers.

3.2 Technologies Required

3.2.1 Relevant Software

This section includes a table that covers the relevant software that will be used to develop the swarm robotic system simulator as well as the planning document. The purpose and description of the software are explained in detail to verify its role and importance to the project as shown in Table 3.

Table 3 Description of relevant software.

Software	Purpose	Description
Python	Development of back-end logic of simulator	*Python* is a general-purpose programming language for building software.
Pycharm	Platform for developing simulator	*Pycharm* is an integrated development environment (IDE), a software that provides tools and a convenient environment for development of software.
Matplotlib	Visualization through plotting of graphs	*Matplotlib* is a library in Python for the visualization of plots.
PyQt5	Visualization through graphical user interface for simulator	*PyQt5* is a *Python* binding of the open-source software Qt which provides widgets for creating graphical user interface.
Figma	Designing of wireframe system	*Figma* is a web application designing tool that allows users to create sketches, prototypes, and so on.
Visual Paradigm	Designing of UML diagrams	*Visual Paradigm* is a software for the creation of various diagrams such as Unified Modelling Language (UML) diagrams, Entity-Relationship Diagram (ERD), and others.
Microsoft Word	Planning and documentation of capstone project	*Microsoft Word* is a word processing software used for documentations.

Microsoft Excel	Creating work plan using Gantt chart	*Microsoft Excel* is a spreadsheet software used to organize and format data.

3.2.2 Hardware Specifications

The following hardware specifications are used to develop, test, and deploy the swarm robotics system simulator project as shown in Table 4.

Table 4 Hardware specifications

Hardware	Specification
Operating system	Microsoft Windows 10 Home
Processor (CPU)	Intel(R) Core(TM) i5-10400F CPU @ 2.90GHz, 2904 Mhz, 6 Core(s), 12 Logical Processor(s)
Installed RAM	16.0 GB 2666 MHz Dual DDR4
Graphics processing unit (GPU)	NVIDIA GeForce RTX 2060

3.3 Software Requirements

3.3.1 Functional Requirements

The proposed functional requirements of the system are as follows:

1. The system should run simulations according to the specified task.
2. The system should run simulations in different types of environments.
3. The system should accept inputs of algorithm scripts for simulation by drag and drop.
4. The system should generate outputs plots for performance analysis of a simulation.
5. The system should allow modification of parameters of the swarm.
6. The system should allow replication of previous simulations.
7. The system should allow recording of a running simulation.
8. The system should provide a 2D visualization interface of a running simulation.
9. The swarm robots in the system should exhibit swarm behaviours.

3.3.2 Non-functional Requirements

The proposed non-functional requirements of the system are as follows:

1. The system should run on the Windows 10 operating system.
2. The system should run simulations as close to real-time.
3. The system should allow control of simulation speed.
4. The system should be able to run simulations on the available hardware.
5. The system should be able handle simulations of up to 100 robots.
6. The system should be able to handle simulations in a large area.
7. The system should generate outputs of the simulation within 10 seconds.
8. The system should be available in the English language.

3.4 Software Architecture

3.4.1 Simulator Design

3.4.1.1 Architecture Design

The model-view-controller (MVC) architecture is implemented in the development of the swarm robotic simulator as shown in Figure 11. The architecture separates the system into three different components, i.e., the model, view, and controller. The model component handles the data logic that is relevant to the user such as from algorithms and parameters. The view component handles the visualizations of the system such as the user interface and only interacts with the controller component to send inputs from interactions with the user interface such as changes of parameters. The controller component acts as an intermediary between the model and view components. It handles requests from the view component such as parameter changes and sends it to the model component to evaluate the parameters and updates the output through the view component to be shown to the user. The controller component can also update the view component directly for minor operations.

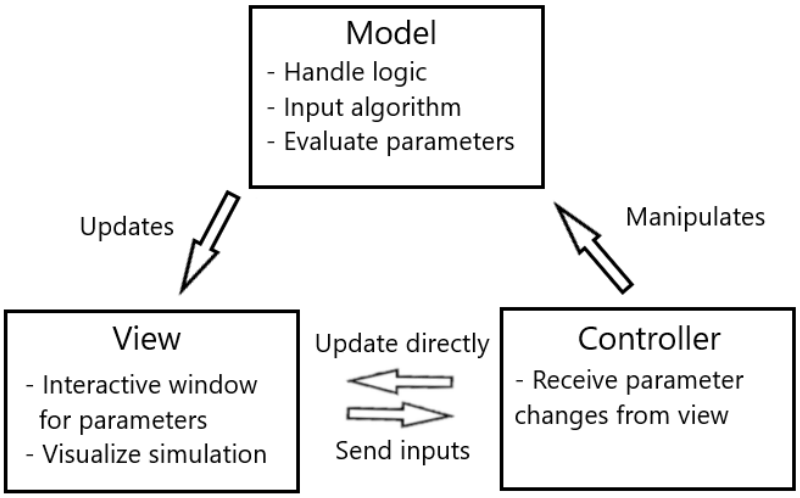


Figure 11 Model-view-controller (MVC) architecture of proposed simulator.

3.4.1.2 Use Case Diagram for User's Perspective

The use case diagram is created to provide an overview on the roles and relationships of the user with the swarm robotic simulator as shown in Figure 12.

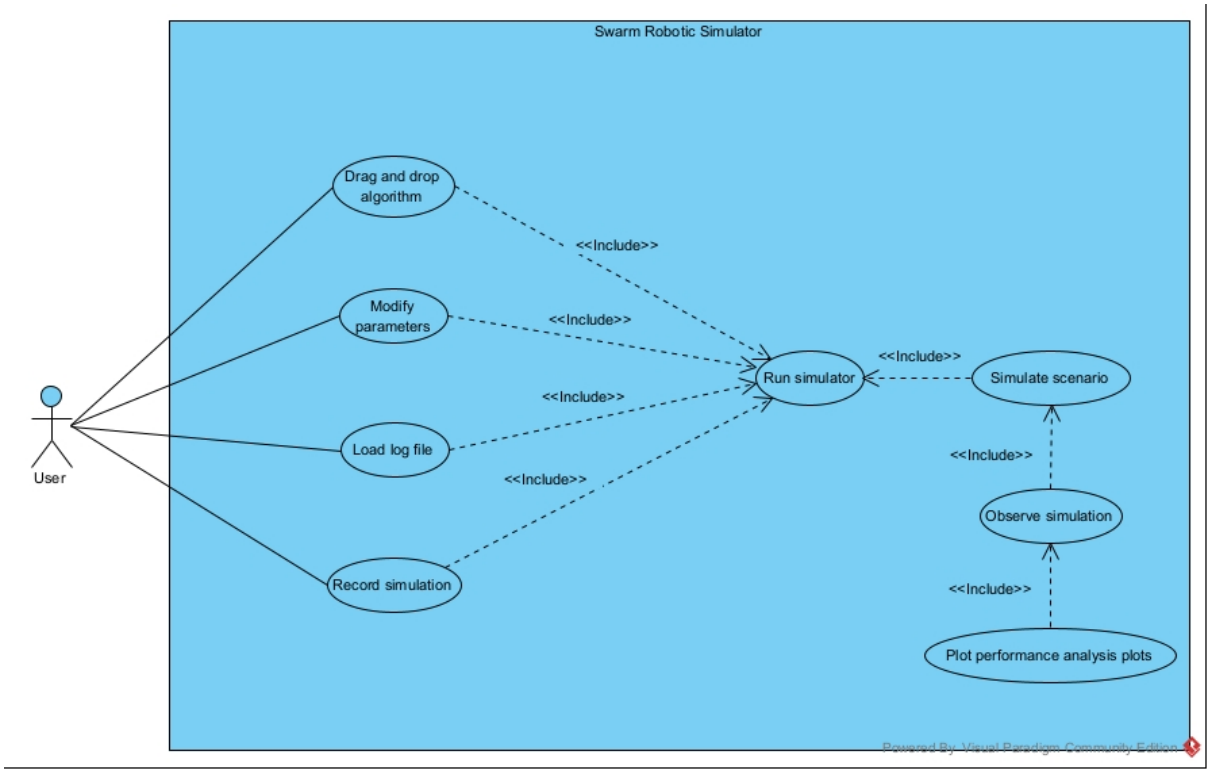


Figure 12 Use case diagram from user's perspective.

3.4.1.3 Prototype of Simulator

A low fidelity prototype of the system is created to visualize the expected structure and design of the system before its implementation as illustrated in Figure 13. The simulator is expected to be fully written in *Python* with the integration of *PyQt5*, a graphical user interface library for *Python* and *Matplotlib*, a *Python* plotting library.

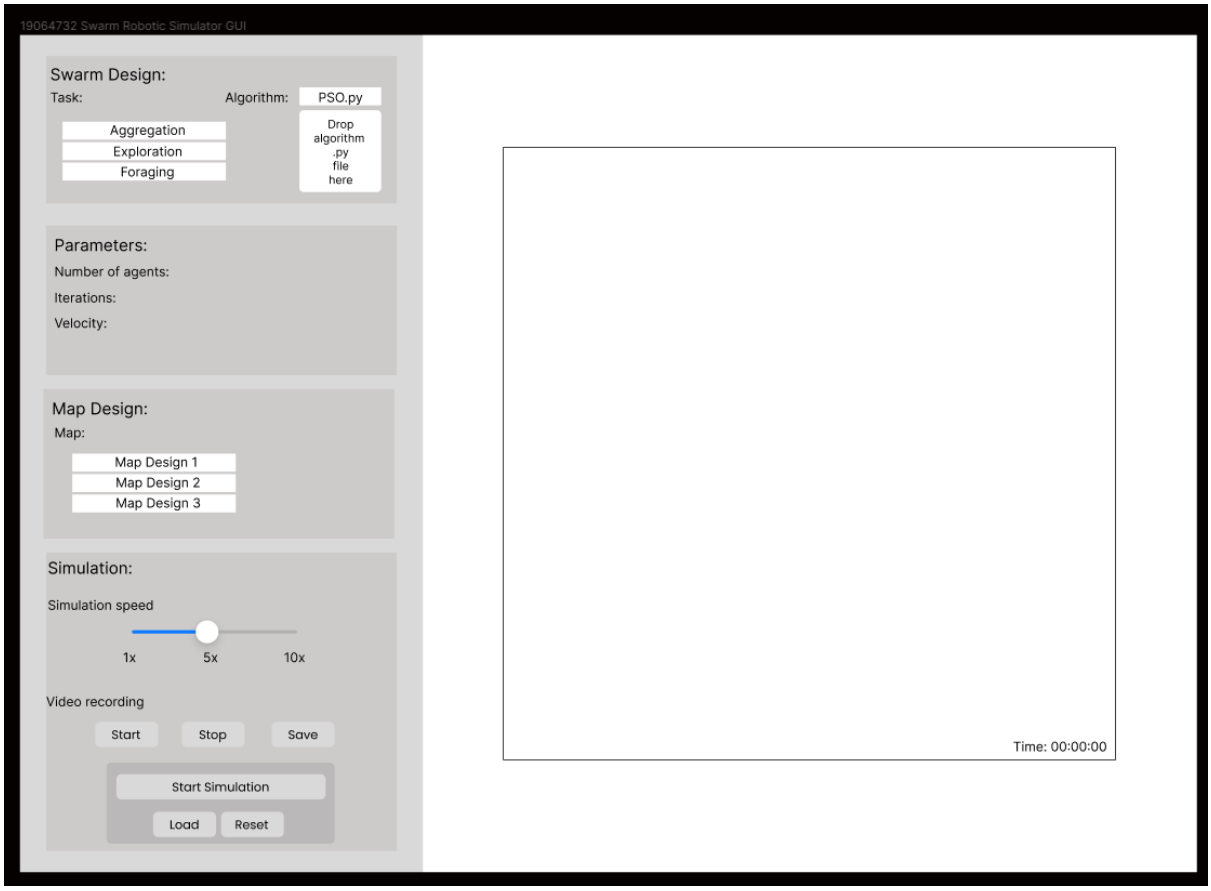


Figure 13 Wireframe of proposed simulator.

3.4.1.4 Simulator Workflow

The simulation workflow is shown in Figure 14. The simulation workflow consists of three steps. It begins with the user modifying the controller parameters which allows users to create different scenarios for the swarm simulation. The controller parameters include the swarm design, agent parameters, and simulation parameters. Otherwise, the user can also load parameters using the load component to load previously run simulations based on saved log files. When the user starts the simulation, the graphical user interface displays the simulation of the swarm to the user. Users can observe issues in the swarm design when the simulation displays the robots attempting to complete an assigned task. During a simulation, the simulation

can be recorded by the user by interacting with the record component which saves a recording in *mp4* format, and the simulation speed can be adjusted by adjusting the slider component for various speed. After the simulation ends, the performance analysis plots are generated and displayed to the user for debugging purposes for the swarm.

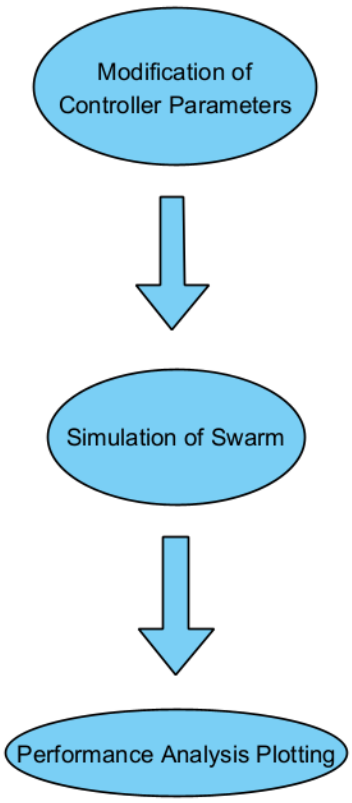


Figure 14 Simulator workflow.

3.4.1.5 Controller Parameters

The simulator provides the user the option to modify the parameters of the simulation and the swarm. In terms of simulation, the simulation speed can be modified from a range of 1 to 10 with 1 as real-time speed and 10 as 10 times faster than real-time. The number of agents, iterations, and velocity of the swarm can also be modified. The simulator should support the number of agents for up to 100 robots. The number of iterations will not have a limit; however, computational runtime of the simulation increases as the number of iterations increase. The velocity of the swarm can also be modified for up to 60 km/h.

3.4.1.6 Accepting and Applying Algorithm

A feature to conveniently accept algorithms from the user is implemented. The feature allows users to drag and drop an algorithm file in the *.py* format from their local system files into the drag/drop component of the simulator to apply the algorithm onto the swarm. This can be done by using the *PyQt5* library which supports for handling drag and drop events. A drop function is defined to handle drop events. The function runs when a file is dropped onto the component which then checks local file for the file path of the algorithm with *.toLocalFile()* using the contents of *.mimeData().urls()*, makes a copy and renames the file, and applies it to the swarm design.

3.4.1.7 Generating Outputs

The simulator can generate plots for the performance analysis and debugging of the swarm. The *matplotlib* library is utilised to generate figures and plots of the swarm performance using the functions, *.figure()* and *.plot()*. Plots such as velocity of the swarm and the performance metrics against time can be generated.

3.4.1.8 Replication of Simulations

The simulator supports the feature of being able to replicate previously run simulations. This can be done by loading in any saved log file which contains the details of the simulator parameters. The log file is generated after every simulation is complete. By clicking on the load component in the user interface, the folder of log files on the local system is displayed. The user must double-click any log file to load the previous simulation parameters into the simulator.

3.4.1.9 Map Designs

Multiple map designs are provided by the simulator for different levels of task difficulty. The first map design will be a small-sized rectangular area with 5 obstacles and 2 scattered items. The second map design will be a medium-sized rectangular area with 10 obstacles and 4 scattered items. The final map design will be large-sized rectangular area with 20 obstacles and 8 scattered items.

3.4.2 Swarm Design

3.4.2.1 Swarm Model

The swarm model represents the robots of a swarm in the simulation used to perform a specific task. A Robot class has the parameters of current coordinates vector, velocity of swarm, graphic variables for visualization, and path planning variables which contains a list of paths. To illustrate, a robot instance can have the parameters of Robot ([50, 50], 40, state, ([50,100], [25, 100], [75, 100])). The robot class contains functions to navigate the map such as *turnLeft()*, *turnRight()*, *goStraight()*, *turnAround()* and *detectWall()*. These functions are necessary for the robot to navigate around the map.

Furthermore, a Swarm class contains the parameters of number of agents, vector of robot objects, and selected algorithm. For instance, a swarm instance can possess the parameters of Swarm (50, robot, PSO).

3.4.2.2 Swarm Algorithms (with flowchart)

3.4.2.2.1 Exploration Algorithm

The exploration algorithm as proposed by [29] is adopted in this simulator. The algorithm consists of three main states which are random walking, obstacle detection, and target detection. In random walking state, the robot travels around the area with consistent speed and constantly scans for nearby objects such as items or walls. In the obstacle detection state, the robot reduces its speed and changes its direction when it detects an obstacle and goes back into random walking state. In target detection stage, the robot stops and investigates any targets and saves the target's relative position and continues with the random walking state.

3.4.2.2.2 Coordinated Depth First Search Algorithm

Another algorithm adopted in the simulator is the coordinated depth first search algorithm proposed by [1]. The process of the algorithm is as follows:

1. Initialization of direction and initial point.
2. Append initial point in a path stack.
3. The algorithm is ended if the path stack is empty, else go to step 4.
4. Detect nearby walls and update local map. The local map information is sent to all robots.

5. Continuously check if there is a nearby point in all directions that has not been visited and without a wall blocking the path. If true, push current point into the path stack and navigate to the nearby point and back to step 3. If false, pop one point from the path stack and go to step 4 and then step 3.

3.4.2.3 Use Case Diagram for Agent

The use case diagram is created to give a summary of the actions of an agent in the simulation environment during a simulation according to the specified algorithm as shown in Figure 15.

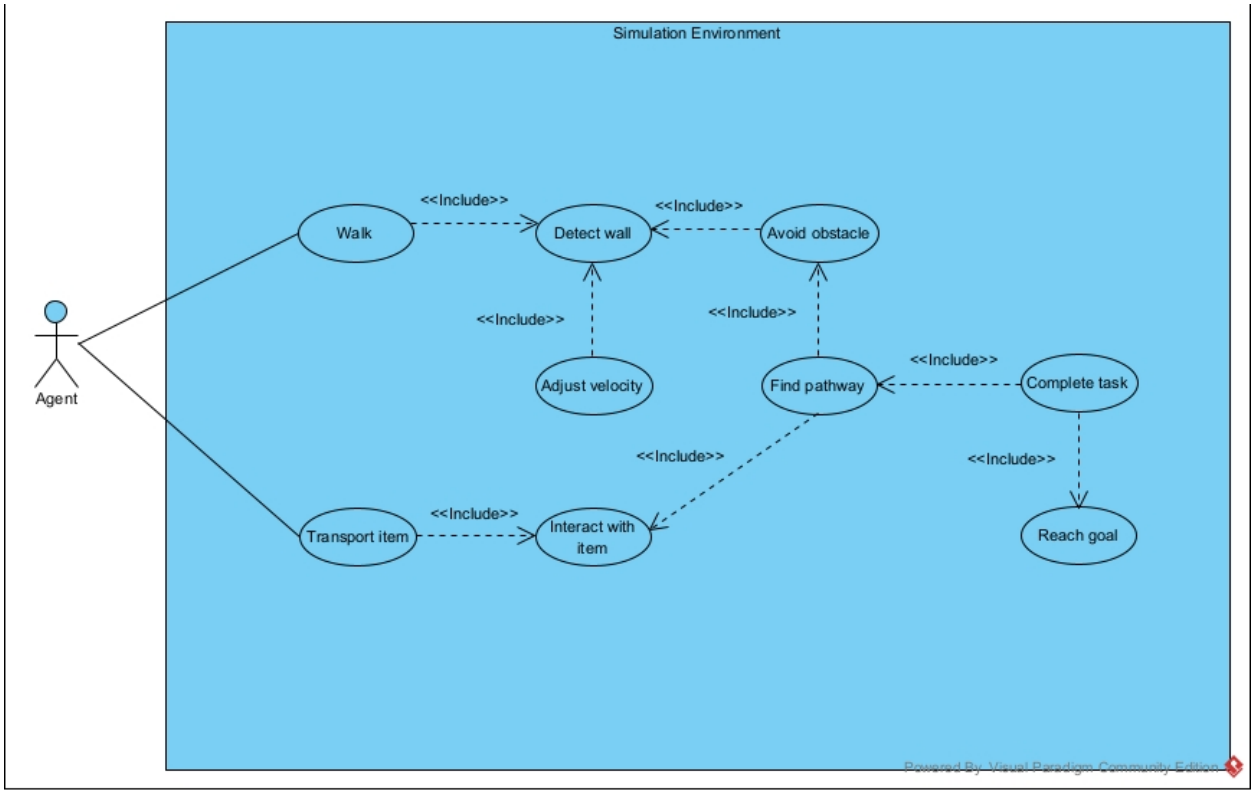


Figure 15 Use case diagram for agent.

3.4.2.4 Swarm Tasks

The main swarm task for this simulator will be navigation tasks which consists of coordinated exploration, coordinated motion, and collective transport. Tasks such as exploration, aggregation, and foraging will be available on the simulator.

In exploration, the swarm must navigate the unexplored map and avoid obstacles to search for the goal location. The task is completed when all the agents in the swarm reaches the goal location.

In aggregation, the swarm will be randomly scattered in the map and an optimum spot is placed on the map. The swarm will have to aggregate, search, and gather at the optimum spot to achieve the task.

In foraging, the swarm will have to search the map for randomly scattered items, pick up the item, and transport the item to the desired location. The task is completed when all the items around the map are picked up and transport to the target location.

3.4.2.5 Swarm Performance Metrics

3.4.2.5.1 Collective Exploration

The performance metrics used for collective exploration will be the ratio of travelled time and the size of robot swarm to reach the target location as recommended by [56, 57] for swarm robotics. To achieve good performance, robots will have to identify the most optimal path to travel from initial point to the goal. For instance, if the travelled time for all the robots to reach target location is 120 seconds, and the swarm size is 100, the performance metrics value will be 120 divided by 100 which is 1.2. The lower the value of performance metrics, the better the performance of the swarm and vice versa.

3.4.2.5.2 Coordinated Motion

The performance metrics for coordinated motion will be the average velocity of the swarm as suggested by [58]. A higher average swarm velocity indicates that the swarm of robots efficiently travels smoothly across the terrain whereas a lower average swarm velocity indicates that the swarm of robots has difficulties navigating the terrain. For instance, a swarm with an average velocity of 50 km/h is said to perform coordinated motion better than a swarm with an average velocity of 10 km/h.

3.4.2.5.3 Collective Transport

The performance metrics for collective transport will be the distance travelled to bring the item to the goal location as proposed by [59]. The efficiency of the collective transport task can be determined by using the ratio of shortest distance to complete the task to the total distance travelled by robots to complete the task. For example, the shortest distance to complete a transportation task is 1 km and the distance achieved by the swarm is 5 km to complete the task. Hence, the efficiency of the swarm will be calculated as 0.2 or interpreted as 20% which is extremely low efficiency.

3.5 Software Testing

3.5.1 Test Cases

The test cases for the proposed are illustrated in Table 5. These test cases ensure that the proposed simulator can behave as expected without bugs and ensuring that the functional requirements of the simulator are satisfied.

Table 5 Test cases for proposed simulator.

Test Case ID	Test Scenario	Test Case Justification	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
TC01	The simulator parameters are modified, and the simulation is run.	To determine if different parameters can be applied to the simulation.	1. Run simulator. 2. Set number of agents to 5. 3. Set number of iterations to 10. 4. Set velocity to 10 km/h. 5. Set task to exploration. 6. Set algorithm to DFS. 7. Run simulation. 8. Observe simulation.	N/A	The parameters of the simulator are reflected in the simulation.		

TC02	The simulation is run with 5 robots at 50 iterations.	To determine if the simulator can run at minimum workload	1. Run simulator. 2. Set number of agents to 5. 3. Set number of iterations to 50. 4. Use default settings for other parameters. 5. Run simulation.	- Number of agents set to 5. - Number of iterations set to 50	The simulator runs without crashing.		
TC03	The simulation is run with 100 robots at 300 iterations.	To determine if the simulator can run at high workload.	1. Run simulator. 2. Set number of agents to 100. 3. Set number of iterations to 300. 4. Use default settings for other parameters. 5. Run simulation.	- Number of agents set to 100 - Number of iterations set to 300	The simulator runs without crashing		
TC04	The user can drag and drop algorithms into the drop component and	To determine if the drag/drop component of the simulator works as intended.	1. Run simulator. 2. Drag and drop algorithm file from local files into drag/drop	- Algorithm file stored in local files.	The algorithm is applied on the swarm during simulation.		

	apply the algorithm onto the swarm design.		component of simulator. 3. Use default settings for other parameters. 4. Run simulation.				
TC05	The user can record a simulation.	To determine if the record component of the simulator works as intended.	1. Run simulator. 2. Use default settings for parameters. 3. Run simulation. 4. Click on the start recording button. 5. Click on the stop recording button when simulation is over. 6. Click on save recording button. 7. Check local files for saved recordings in recording folder.	N/A	A recording of the simulation is saved into local files in *mp4* format.		

TC06	The user can adjust the speed of a running simulation.	To determine if the slider component for speed works as intended.	1. Run simulator. 2. Use default settings for parameters. 3. Run simulation. 4. Adjust simulation speed. 5. Observe simulation speed.	N/A	The simulation speed increases or decreases during simulation.		
TC07	The simulator generates a log file after simulation ends.	To determine if the log file for replication of simulations can be generated.	1. Run simulator. 2. Use default settings for parameters. 3. Run simulation. 4. Check log folder for log file.		A log file from the simulation should be generated in the log folder.		
TC08	The user can load previously saved log files to edit controller parameters and replicate simulations.	To determine if the load component of the simulator works as intended.	1. Run simulator. 2. Click on the load component. 3. Select a log file from folder. 4. Run simulation.	- Existing log file in log folder	The simulation runs according to the parameters of the log file.		

4 Work Plan

4.1 Gantt Chart

A Gantt chart which consists of a list of tasks and the schedule to complete these tasks is designed as shown in Figure 16. The aim of the Gantt chart is to assist in the time management of the project and ensure the success of the project.

Project: Capstone Project 1
Topic: Research-centric simulator for swarm robotic system
Date: 22/08/2022 - 2/12/2022

Section	Content	Task	Week														Duration (weeks)
			1	2	3	4	5	6	7	8	9	10	11	12	13	14	
Preliminary	Topic and Supervisor	Briefing on Capstone Project 1															2
		Decision on topic and supervisor															
Documentation	Work Plan	Preparation and Finalization of Gantt Chart															3
	Introduction	Preparation of Introduction draft															
		Finalization of Introduction															
	Literature Review	Preparation of Literature Review draft															4
		Finalization of Literature Review															
	Technical Plan	Preparation of Technical Plan draft															4
		Finalization of Technical Plan															
	Planning Document	Review and Finalization of Planning Document															1
	Activity Log	Completion of Timeline, Bibliography, Meeting Records															-
Submission	Submission	Submission of Planning Document and Activity Log															-
																Total:	14

Project: Capstone Project 2
Topic: Research-centric simulator for swarm robotic system
Date: 27/04/2023 - 04/08/2023

Section	Content	Task	Week														Duration (weeks)
			1	2	3	4	5	6	7	8	9	10	11	12	13	14	
Implementation	Base GUI	Develop the basic graphical user interface of simulator															1
	Simulator Logic	Develop the simulator logic of simulator															2
	Swarm Logic	Develop the swarm logic of simulator															2
	Swarm and Simulator Logic	Integrate both swarm and simulator logic into the simulator															3
Test and Debug	Simulator Functionalities	Test and debug the simulator functionalities according to test cases															1
Documentation	Results and Discussion	Prepare and finalize results and discussion															1
	Revision of CP1 Report	Proofread and revise the CP1 report and make changes															
	Conclusion	Prepare and finalize conclusion															1
	Revision of CP2 Report	Proofread and revise the CP2 report and make changes															
Submission	Submission	Submission of Final Report and Project Files															1
Presentation	Presentation Slides	Prepare presentation slides															
	Presentation and Demonstration	Presentation and Demonstration of Capstone Project															2
																Total:	14

Figure 16 Gantt chart for capstone project.

4.1.1 Description of Tasks in Gantt Chart

4.1.1.1 Capstone Project 1

1. Briefing on capstone project 1 (Week 1)

   The first week consists of a briefing on the capstone project 1 which is provided by the course coordinator. One week allocated in this task is fixed because it depends on the course coordinator.

2. Decision on topic and supervisor (Week 2)

   On the second week, the final year student must decide on a topic and a supervisor for the capstone project. One week allocated in this task is fixed because it depends on the course coordinator.

3. Preparation and finalization of Gantt chart (Week 3 – Week 4)

   On the third and fourth week, the Gantt chart must be completed for planning purposes. Two weeks are allocated in this task because it is done simultaneously with task 4.

4. Preparation and finalization of introduction (Week 3 – Week 5)

   On the third to fifth week, the introduction section of the report must be completed. Three weeks are allocated in this task because it is done simultaneously with task 3 and requires lots of research to a new topic.

5. Preparation and finalization of literature review (Week 6 – Week 9)

   On the sixth to ninth week, the literature review section of the report must be completed. Four weeks are allocated in this task because it requires a lot of research to the new topic and a lot of time is required to gather, compile, and paraphrase the content.

6. Preparation and finalization of technical plan (Week 10 – Week 13)

   On the tenth to thirteenth week, the technical plan section of the report must be completed. Four weeks are allocated in this task because it requires the application and deep understanding of content in literature review.

7. Review and finalization of planning document (Week 14)

   On the last week, the planning document which consists of all the previous sections, are proofread, and reviewed. One week is allocated in this task because it only summarises up the planning document.

8. Completion of timeline, bibliography, meeting records (Week 2, Week 4, Week 6, Week 8, Week 10, Week 13)

   The timeline and activities, bibliography and annotations, and meeting records are done throughout 14 weeks when meeting with the supervisor are held. Multiple weeks are allocated here because the meeting records need to be done bi-weekly.

9. Submission of planning document and activity log (Week 14)

   The complete planning document and activity log are submitted in week 14 for evaluation.

4.1.1.2 Capstone Project 2

1. Develop the basic graphical user interface of simulator (Week 1)

   In week 1, the basic graphical user interface of the simulator is developed using *PyQt5* library. One week is allocated in this task because the development of the GUI can be based off the drafted wireframe.

2. Develop the simulator logic of simulator (Week 2 – Week 3)

   In week 2 to 3, the simulator logic is implemented such as its base functionalities. Two weeks are allocated because the content of the simulator logic may take time to learn and apply.

3. Develop the swarm logic of simulator (Week 4 – Week 5)

   In week 4 to 5, the swarm logic is developed such as the swarm models and algorithms. Two weeks are allocated because the content of the swarm logic may take time to learn and apply.

4. Integrate both swarm and simulator logic into the simulator (Week 6 – Week 8)

   In week 6 to 8, both the swarm and simulator logic are integrated to develop a complete simulator. Three weeks are allocated because the integration of both parts can cause many bugs.

5. Test and debug the simulator functionalities according to test cases (Week 9)

   In week 9, the complete simulator is tested according to the planned test cases and debugged where necessary. One week is allocated because the functionalities of the simulator are not too complex.

6. Prepare and finalize results and discussion (Week 10)

   In week 10, the results and discussion sections of the report are written. One week is allocated because results and discussion can be written off the software testing section.

7. Proofread and revise the CP1 report and make changes (Week 10)
   In week 10, the previous capstone project 1 report is revised and made changes where necessary. One week is allocated here together with task 6 because the report may require some changes.

8. Prepare and finalize conclusion (Week 11)
   In week 11, the conclusion section of the report is prepared and finalized. One week is allocated here because conclusion will only require summarising the whole report.

9. Proofread and revise the CP2 report and make changes (Week 11)
   In week 11, the complete capstone project 2 report is proofread and revised, and changes are made where necessary. One week allocated here together with task 8 because proofreading will not take too much time.

10. Submission of Final Report and Project Files (Week 12)
    In week 12, the final report and project files of the capstone project are submitted for evaluation.

11. Prepare presentation slides (Week 12)
    In week 12, the presentation slides are prepared for the upcoming presentation.

12. Presentation and Demonstration of Capstone Project (Week 12 – Week 14)
    From week 12 to 14, a presentation slot is booked, and the capstone project is presented to the respective examiners.

4.2 Project Deliverables

The project deliverables for the capstone project are shown in Table 6.

Table 6 Project deliverables.

Capstone	No.	Deliverable Name
1	1	Planning document
	2	Activity log
2	3	Project report
	4	Project files
	5	Presentation slides

4.3 Project Risk Factors

The project risk factors are evaluated and discussed in Table 7 to determine possible threats to the successful development of the project and develop solutions to issues that may arise during the execution of the project.

Table 7 Project risk factors.

No.	Risk Definition	Probability	Impact	Mitigation
1	Scope covered is too broad	Low	Medium	- Consult supervisor and redefine a more realistic scope for the project
2	Low productivity from burnout	High	Low	- Take frequent breaks instead of working for long continuous sessions
3	Unable to learn and implement new libraries	Medium	High	- Consult supervisor for guidance

				- Perform further research on the libraries - Introduce new libraries for implementation under supervisor's approval
4	Tight deadlines	Low	High	- Reintroduce a work plan with more realistic deadlines and allocate more time into the project
5	Developed simulator has too many bugs	High	Medium	- Test codes more frequently and resolve bugs when errors arise - Introduce more test cases and perform debugging

References

[1] J. Song, A. Song, and F. Zhang, "RoboMaze: SWARM robotics and coordinated navigation in Smart City," *2021 IEEE 23rd Int Conf on High Performance Computing & Communications; 7th Int Conf on Data Science & Systems; 19th Int Conf on Smart City; 7th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*, 2021.

[2] Khaldi, B.; Cherif, F. An overview of swarm robotics: Swarm intelligence applied to multi-robotics. Int. J. Comput. Appl. 2015, 126, 31–37.

[3] A. Liekna and J. Grundspenkis, "Towards practical application of swarm robotics: Overview of swarm tasks," *Engineering for rural development*, May 2014.

[4] J. C. S de Sá, "RViz Based Simulation For Motion Control Of Swarm Robotics," 2022.

[5] P. S. Andrews, S. Stepney, and J. Timmis, "Simulation as a scientific instrument," *Proceedings of the 2012 workshop on complex systems modelling and simulation*, 2012.

[6] N. Nedjah and L. S. Junior, "Review of methodologies and tasks in swarm robotics towards standardization," *Swarm and Evolutionary Computation*, vol. 50, p. 100565, 2019.

[7] A. R. Cheraghi, S.Shahzad, and K. Graffi, "Past, present, and future of swarm robotics," In *Proceedings of SAI Intelligent Systems Conference* (pp. 190-233). Springer, Cham, 2021.

[8] J. C. Barca and Y. A. Sekercioglu, "Swarm robotics reviewed," Robotica, vol. 31, no. 3, pp. 345–359, 2013.

[9] G. Beni, "The concept of cellular robotic system," in Proceedings IEEE International Symposium on Intelligent Control 1988, Aug 1988, pp. 57–62.

[10] T. Fukuda and S. Nakagawa, "Approach to the dynamically reconfigurable robotic system," Journal of Intelligent and Robotic Systems, vol. 1, no. 1, pp. 55–72, 1988.

[11] G. Beni, and J. Wang, “Swarm Intelligence in Cellular Robotic Systems. Robots and biological systems: towards new bionics,” pp. 703--712. Berlin, Springer, 1993.

[12] G. Beni, “From swarm intelligence to swarm robotics,” in Swarm Robotics Workshop: State-of-the-Art Survey, E. Şahin and W. Spears, Eds., no. 3342, pp. 1–9, Springer, Berlin, Germany, 2005.

[13] G. Beni and J. Wang, “Swarm intelligence,” In Proc. of the Seventh Annual Meeting of the, pages 425–428, Tokyo, Japan, 1989.

[14] E. Osaba, J. Del Ser, A. Iglesias, and X.-S. Yang, “Soft computing for swarm robotics: New trends and applications,” *Journal of Computational Science*, vol. 39, p. 101049, 2020.

[15] D. P. Cruz, R. D. Maia, and L. N. De Castro, “A critical discussion into the core of swarm intelligence algorithms,” Evolutionary Intelligence, 12(2), 189—200, 2019.’

[16] Dorigo M., Birattari M. 2007. Swarm Intelligence. Scholarpedia. 2(9), 2007.

[17] X.-S. Yang, S. Deb, Y.X. Zhao, S. Fong, X.S. He, Swarm intelligence: past, present and future, Soft Computing, 10.1007/s00500-017-2810-5, 2017.

[18] R. Eberhart, P. Simpson, and R. Dobbins, Computational intelligence PC tools. Academic Press Professional, Inc., 1996.

[19] R. Eberhart and J. Kennedy, “A new optimizer using particle swarm theory,” in Micro Machine and Human Science, 1995. MHS’95., Proceedings of the Sixth International Symposium on. IEEE, 1995, pp. 39–43.

[20] J. Kennedy, “The particle swarm: social adaptation of knowledge,” in Evolutionary Computation, 1997., IEEE International Conference on. IEEE, 1997, pp. 303–308.

[21] Mehdi N., Ghodrat S., Mehdi S., Adel N. T. Artificial fish swarm algorithm: a survey of the state of the-art, hybridization, combinatorial and indicative applications. Artif Intell Rev. Springer Science and Business Media B.V. 2012. 10.1007/s10462-012- 9342-2, 2012.

[22] Hamdi A., Monmarché N., Adel Alimi M., Slimane M. 2011. Bee-based algorithms: a review. http://www.researchgate.net/publication/236341293_Beebased_algorithms_a_review/file/72e7e51bea05a01b3c.pdf.

[23] M. Dorigo and E. Sahin. Special issue: Swarm robotics. Autonomous Robots, 17:111–113, 2004.

[24] E. Sahin. Swarm robotics: From sources of inspiration to domains of application. In E. Sahin and William Spears, editors, Swarm Robotics Workshop: State-ofthe-art Survey, number 3342 in Lecture Notes in Computer Science, pages 10–20, Berlin Heidelberg, 2005. Springer-Verlag.

[25] S. Camazine, J.-L. Deneubourg, N.R. Franks, J. Sneyd, G. Theraulaz, and E. Bonabeau. Self-Organisation in Biological Systems. Princeton University Press, NJ, USA, 2001.

[26] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, "Swarm robotics: a review from the swarm engineering perspective," Swarm Intelligence, vol. 7, no. 1, pp. 1–41, 2013.

[27] E. Şahin, S. Girgin, L. Bayindir, and A. E. Turgut, “Swarm Robotics,” *Natural Computing Series*, 2008, pp. 87–100.

[28] B. Septfons, A. Chehri, H. Chaibi, R. Saadane, and S. Tigani, “Swarm robotics: Moving from concept to application,” *Human Centred Intelligent Systems*, pp. 179–189, 2022.

[29] X. Huang, F. Arvin, C. West, S. Watson, and B. Lennox, “Exploration in extreme environments with swarm robotic system,” *2019 IEEE International Conference on Mechatronics (ICM)*, 2019.

[30] Janson, S.; Merkle, D.; Middendorf, M. A decentralization approach for swarm intelligence algorithms in networks applied to multi swarm PSO. Int. J. Intell. Comput. Cybern. 2008, 1, 25-45.

[31] S. Garnier, J. Gautrais, and G. Theraulaz, “The biological principles of swarm intelligence,” Swarm Intelligence, vol. 1, no. 1, pp. 3–31, Jun 2007. [Online]. Available: https://doi.org/10.1007/s11721-007-0004-y

[32] P. G. Dias, M. C. Silva, G. P. Rocha Filho, P. A. Vargas, L. P. Cota, and G. Pessin, “Swarm robotics: A perspective on the latest reviewed concepts and applications,” *Sensors*, vol. 21, no. 6, p. 2062, 2021.

[33] Tom De Wolf and Tom Holvoet. Design patterns for decentralised coordination in self-organising emergent systems. In International Workshop on Engineering Self-Organising Applications. Springer, 28–49, 2006.

[34] Z. Sun, “Robot swarm navigation: Methods, analysis, and applications,” *2021 2nd International Conference on Big Data & Artificial Intelligence & Software Engineering (ICBASE)*, 2021.

[35] Liekna, A., & Grundspenkis, J. Towards practical application of swarm robotics: overview of swarm tasks. Engineering for rural development, 13, 271-277, 2014.

[36] R. Groß and M. Dorigo, "Towards group transport by swarms of robots," International Journal of BioInspired Computation, vol. 1, no. 1-2, pp. 1–13, 2009.

[37] M. Hiraga, Y. Wei, and K. Ohkura, "Evolving collective cognition for object identification in foraging robotic swarms," Artificial Life and Robotics, vol. 26, no. 1, pp. 21–28, 2021.

[38] Maxim P. M., Spears W. M., Spears D. F. Robotic chain formations. In Proceedings of the IFAC. workshop on networked robotic (NetRob'09).19–24, 2009.

[39] Hamann, H. Swarm Robotics: A Formal Approach; Springer: Cham, Switzerland, 2018.

[40] F. Arvin, A. E. Turgut, F. Bazyari, K. B. Arikan, N. Bellotto, and S. Yue, “Cue-based aggregation with a mobile robot swarm: a novel fuzzy-based method,” Adaptive Behavior, vol. 22, no. 3, pp. 189–206, 2014.

[41] V. Trianni, R. Groß, T. H. Labella, E. Sahin, and M. Dorigo, “Evolving aggregation behaviors in a swarm of robots,” in Advances in Artificial Life, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2003, vol. 2801, pp. 865–874.

[42] F. Arvin, A. E. Turgut, T. Krajn´ık, S. Rahimi, E. O. ˙Ilkin, S. Yue, S. Watson, and B. Lennox, “ΦClust: Pheromone-based Aggregation for Robotic Swarms,” in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018, pp. 4288–4294.

[43] O. Soysal and E. Sahin, “Probabilistic aggregation strategies in Swarm Robotic Systems,” *Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005.*

[44] Dorigo M., Trianni V., S¸ahin E., Gross R., Labella T. H., Baldassare G., Nolfi S., Mondada F., Deneubourg J-L., Mondada F., Floreano D., Gambardella L., Evolving Self-Organization Behaviors for a Swarm-bot, Autonomous Robots 17, 223-245, 2004.

[45] Melhuish, C., Welsby, J., Edwards, C.: Using templates for defensive wall building with autonomous mobile antlike robots. In: Proceedings of Towards Intelligent Mobile Robots (TIMR'99), 1999.

[46] Justin, W.: Extended Stigmergy in Collective Construction. In: Radhika, N. (ed.), vol. 21, pp. 20- 28, 2006.

[47] Werfel, J., Petersen, K., Nagpal, R.: Distributed multi-robot algorithms for the TERMES 3D collective construction system. URL http://www.eecs.harvard.edu/ssr/publications/. Last checked on November 2012.

[48] Lee, W.; Vaughan, N.; Kim, D. Task Allocation Into a Foraging Task With a Series of Subtasks in Swarm Robotic System. IEEE Access 2020, 8, 107549–107561.

[49] Jansson, F., Hartley, M., Hinsch, M., Slavkov, I., Carranza, N., Olsson, T. S., ... & Grieneisen, V. A. Kilombo: a Kilobot simulator to enable effective research in swarm robotics. arXiv preprint arXiv:1511.04285, 2015.

[50] Bayındır, L. A review of swarm robotics tasks. Neurocomputing 2016, 172, 292–321.

[51] C. Calderón-Arce, J. C. Brenes-Torres, and R. Solis-Ortega, "Swarm robotics: Simulators, platforms and Applications Review," *Computation*, vol. 10, no. 6, p. 80, 2022.

[52] Erez, T.; Tassa, Y.; Todorov, E. Simulation tools for model-based robotics: Comparison of bullet, havok, mujoco, ode and physx. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 26–30 May 2015; pp. 4397–4404.

[53] A. R. Cheraghi, K. Actun, S. Shahzad, and K. Graffi, "Swarm-SIM: A 2D & 3D simulation core for swarm agents," *2020 3rd International Conference on Intelligent Robotic and Control Engineering (IRCE)*, 2020.

[54] E. Soria, F. Schiano, and D. Floreano, "Swarmlab: A matlab drone swarm simulator," *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[55] S. Lim, S. Wang, B. Lennox, and F. Arvin, "Beeground - an open-source simulation platform for large-scale Swarm Robotics applications," *2021 7th International Conference on Automation, Robotics and Applications (ICARA)*, 2021.

[56] F. Ducatelle et al., "Cooperative navigation in robotic swarms," Swarm Intelligence, vol. 8, no. 1, pp. 1–33, 2014.

[57] M. S. Talamali, T. Bose, M. Haire, X. Xu, J. A. R. Marshall, and A. Reina, "Sophisticated collective foraging with minimalist agents: a swarm robotics test," Swarm Intelligence, vol. 14, no. 1, pp. 25–56, 2020.

[58] A. E. Turgut, H. Çelikkanat, F. Gökçe, and E. Şahin, "Self-organized flocking in mobile robot swarms," Swarm Intelligence, vol. 2, no. 2, pp. 97– 120, 2008.

[59] J. Chen, M. Gauci, and R. Groß, "A strategy for transporting tall objects with a swarm of miniature mobile robots," in 2013 IEEE International Conference on Robotics and Automation, pp. 863– 869.