TEB1113/TFB2023: ALGORITHM & DATA STRUCTURE

September 2024

Group Name: Synchrotech

Homework 4

Prepared for: Dr. Nordin Zakaria

Num.	Full Name	Student ID	Course
1.	Siti Nurfatimah Az Zahra binti Norhisham	20001348	Computer Science
2.	Addly Aiman bin Mohamad Faizal	22004410	Computer Engineering
3.	Nur Husna Husniyah binti Abdul Razi	22002729	
4.	Yasreen bt Mohamed Yusoff	22005648	
5.	Nurul Anisa Binti Sufian	22005637	

TABLE OF CONTENTS

TABLE OF FIGURES

1.0 Application Domain

Device Specification

1. **Model:** Victus 15
2. **RAM:** 16GB
3. **Storage:** 500GB
4. **Processor:** AMD Ryzen 5 7535 HS
5. **GPU:** Radeon Graphics
6. **Operating System:** Windows 11

2.1 Introduction

This project simulates a drone swarm designed for security applications. The drones are programmed to patrol a specified area, changing color as they approach the boundary. If a drone exceeds a defined radius, it self-destructs to prevent straying, and this self-destruction can also be triggered based on the drone's unique Drone ID. The system ensures that the drones move cohesively through key behaviors such as alignment, avoidance, and cohesion. These behaviors allow the drones to maintain their formation and work together autonomously, highlighting the potential for this approach in security applications where coordinated movement and boundary adherence are crucial.

This project extends a drone swarm simulation designed for security applications, introducing a communication network within each partition of drones. The network is modeled as a Binary Search Tree (BST), with each drone linked to one other drone in the partition. This structure facilitates efficient communication and targeted actions, such as sending self-destruct commands to specific drones. Key behaviors like alignment, avoidance, and cohesion ensure coordinated movement within the swarm, while the binary tree-based communication system optimizes directed communication paths.

2.2 Swarm Behavior Algorithms

The core of this simulation is based on algorithms that govern the behaviors of individual drones, ensuring they work together cohesively as a swarm. These algorithms are designed to handle

various aspects of the drone's movement and interaction with others, optimizing the swarm's collective efficiency in security operations.

- Alignment Algorithm: This algorithm ensures that each drone aligns its direction with the average direction of nearby drones, promoting synchronized movement across the swarm. The alignment algorithm's complexity is O(n), where n is the number of drones in the local vicinity.
- Avoidance Algorithm: The avoidance algorithm calculates the distance between drones and ensures that they do not collide with each other. By maintaining a safe distance, drones avoid overlap and movement conflicts, allowing for smooth and efficient operation within the swarm. This algorithm also operates with a time complexity of O(n), where n is the number of drones in the area.
- Cohesion Algorithm: The cohesion algorithm keeps the drones within a certain proximity of each other, encouraging them to stay close to the center of the swarm. This ensures the swarm functions as a unified group rather than scattered drones. The complexity of this algorithm is O(n), considering the need to evaluate each drone's position relative to others.
- Radius Check Algorithm: This algorithm monitors each drone's distance from the designated boundary. If a drone exceeds the set radius, it triggers the self-destruction sequence. Since it only requires calculating the distance between a drone and the center of the patrol area, this algorithm runs in O(1) time for each drone.

Together, these swarm behavior algorithms enable the drones to function as a coordinated unit, responding to environmental changes and maintaining formation. The overall performance of the simulation is optimized by these algorithms, with their complexity ensuring that the system can handle large numbers of drones efficiently.

2.3 Binary Tree Integration

A binary tree data structure is integrated into the simulation to manage drone data and boundary checks efficiently. Each drone is represented as a node in the binary tree, storing key attributes such as its unique Drone ID, position, and current status. This structure allows for fast retrieval, insertion, and deletion of drone information, which is essential for real-time updates during the simulation. Additionally, boundary checks for each drone are optimized using the binary tree, enabling quicker decision-making regarding self-destruction when a drone exceeds its patrol

boundary. The binary tree's time complexity for these operations is O(log n), where n is the number of drones.

2.3.1 DroneBTCommunication Class

The DroneBTCommunication class represents the binary tree communication network within each partition. In this setup:

- Each partition of drones is represented as a separate binary tree, managed by a DroneBTComm object.
- The binary tree structure links drones using a specified attribute (e.g., Drone ID or position) as the key, allowing efficient searches and targeted messaging within the network.

2.3.2 Binary Tree Operations

- Search by Key: Locates a specific drone using a designated key (such as Drone ID) and retrieves its position. Complexity: O(log n) if the tree is balanced.
- Exhaustive Search by Attribute: Performs a complete search based on other attributes. Complexity: O(n), as all nodes are checked.
- Self-Destruct Command: Sends a self-destruct message to a drone, causing it to disappear from the scene (SetActive(false)).

Each communication action considers the simulated Euclidean distance between drones, representing message transfer time. This binary tree-based communication model minimizes message relay hops, enhancing efficiency for large-scale drone swarms.

2.4 Performance Analysis

The performance analysis examines the efficiency of communication and swarm behaviors with varying numbers of drones, assessing frame rate, response times, and search speeds. Key factors include:

- Communication Efficiency: The binary tree structure ensures fast message routing and retrieval operations within each partition. For a balanced tree, the average search and

message relay time complexity is O(log n). Exhaustive searches based on specific attributes (other than the primary key) run in O(n) time.

- Frame Rate Measurement: Frame rate data is collected as the number of drones increases, revealing the impact of larger swarms on system responsiveness. Monitoring frame rate helps determine the optimal number of drones that can be managed without degrading performance.

- Simulated Communication Time: The communication delay is simulated based on Euclidean distance between drones, giving realistic timing for operations like search and self-destruction. Performance results show how the tree structure affects average response time, with expected increases as the swarm size grows.

2.4.1 Impact on Performance Analysis

- CPU: A mid-range processor like the Ryzen 5 should handle the computational load for a moderate-size drone swarm effectively, but as the simulation size grows, performance may be affected. Complex behaviors and large numbers of drones could slow down computation.
- Memory: With 16GB of RAM, the system should perform well under most conditions, but very large swarms or complex data structures (like the binary tree) may strain memory usage, especially if drones' data grows more detailed.
- Storage: As long as disk writing is minimized or handled efficiently, storage capacity and speed won’t be a major concern for real-time operations. If persistent storage or logging is involved, an SSD would be preferred to reduce potential lag.
- GPU: If your simulation includes rendering visual data (like in a Unity-based environment), the GPU will enhance performance. However, for purely algorithmic tasks, the GPU will not be significantly involved.

2.5 Screenshot(s)

Figure 1: Introduction Page

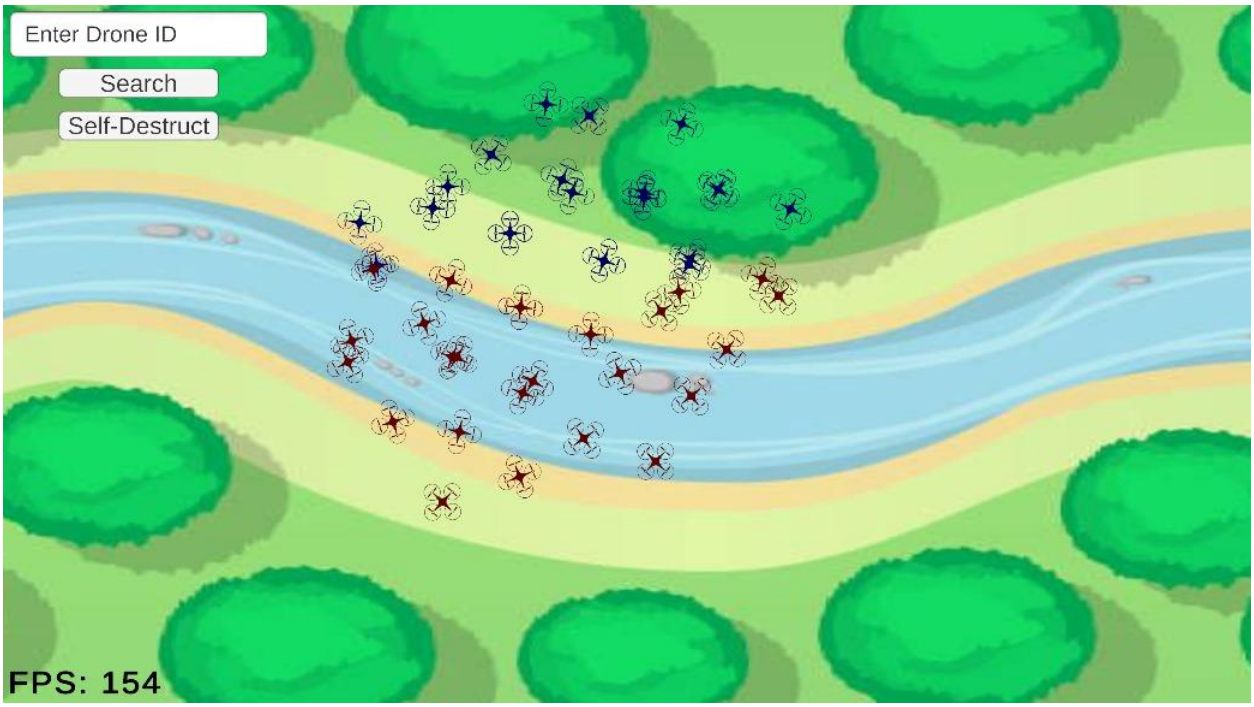

Figure 2: Swarm of Drones

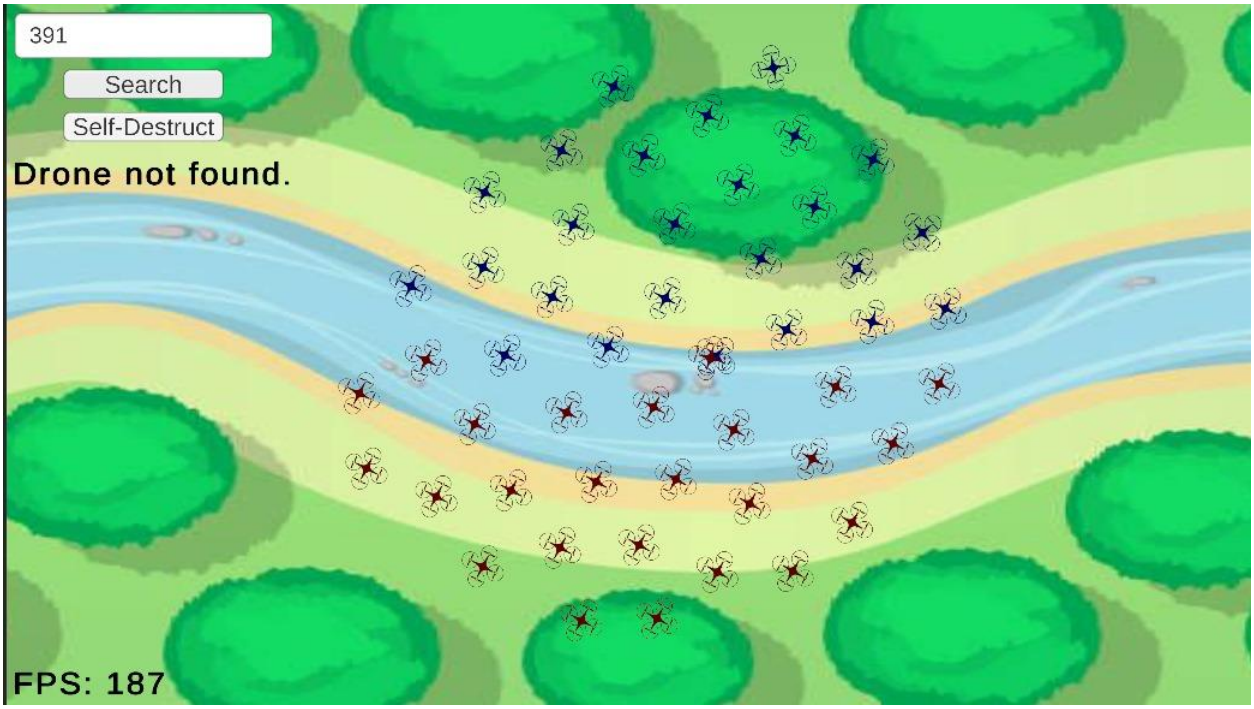

Figure 3: 'Drone ID' not Found

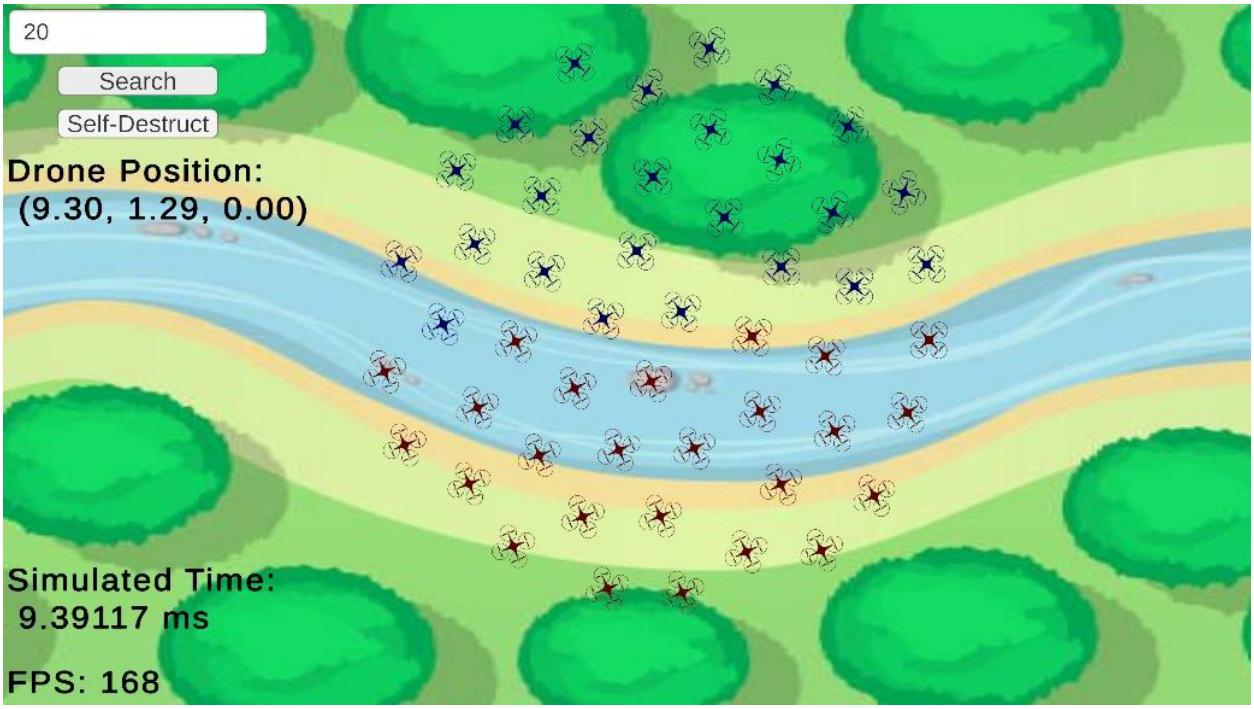

Figure 4: 'Drone Position' found by 'Drone ID'

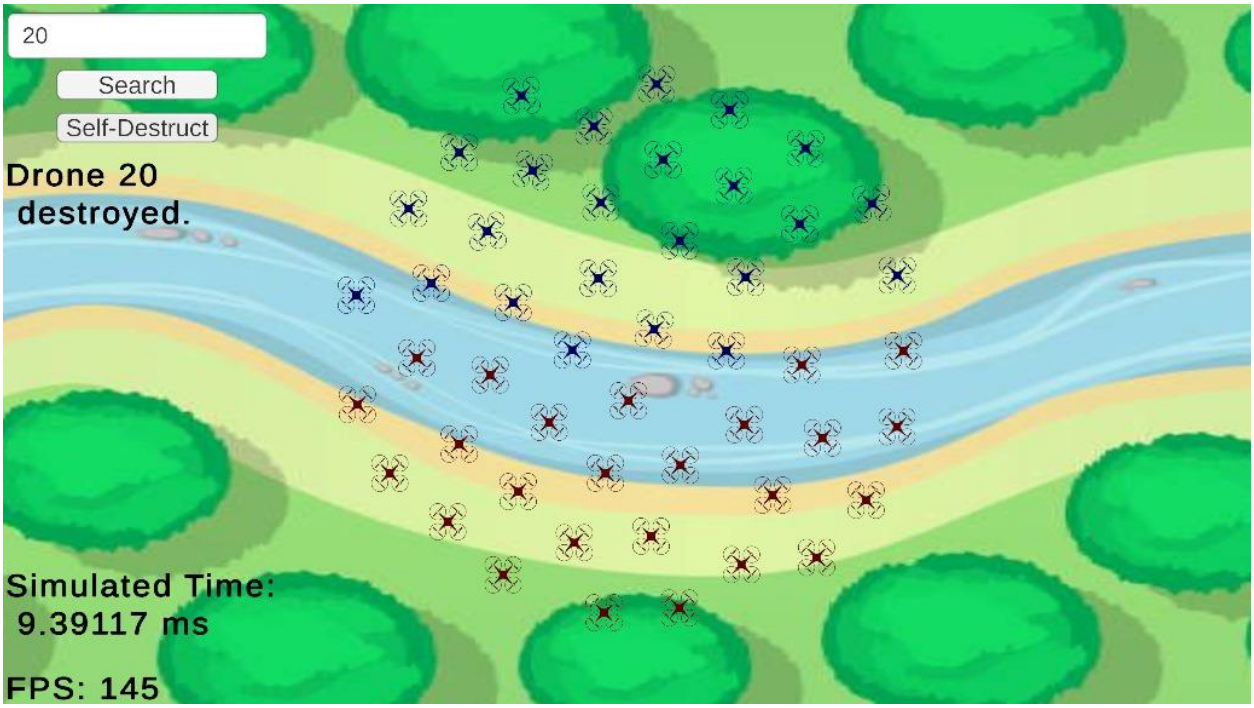

Figure 5: Drone destroyed by 'Drone ID'

2.6 Asset Images

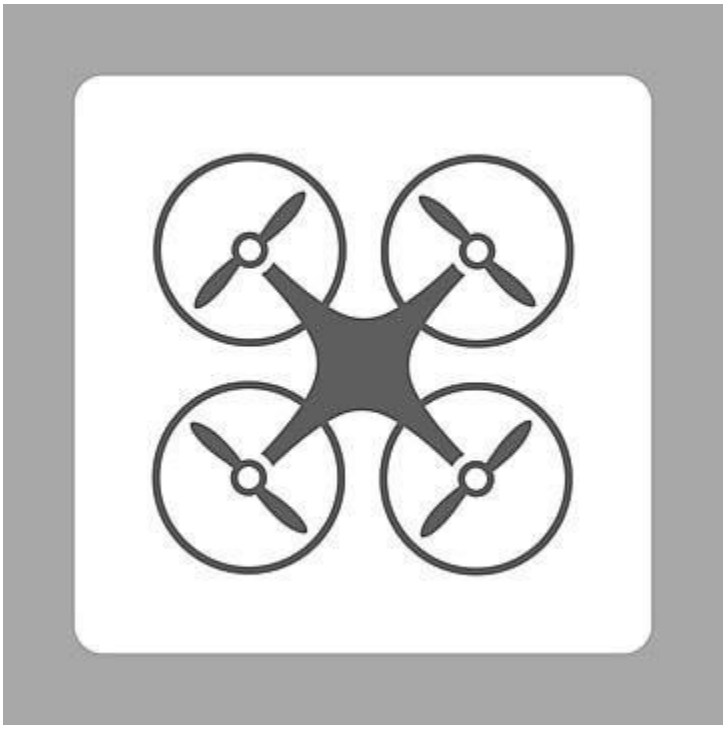

Figure 6: Drone Models

Figure 7: Start Button

Figure 8: Background