МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет

Факультет електроніки та інформаційних технологій

Кафедра комп'ютерних наук

«До захисту допущено»

В.о. завідувача кафедри

_____ Ігор ШЕЛЕХОВ

(підпис)

09 червня 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 – Комп'ютерних наук,

освітньо- професійної програми «Інформатика»

на тему: «Інформаційна система симуляції польоту рою дронів»

здобувача групи ІН-92 Кияненка Дмитра Віталійовича

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Дмитро КИЯНЕНКО

(підпис)

Керівник
доцент,
кандидат технічних наук Сергій ПЕТРОВ _____

(підпис)

Суми – 2023

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавра

зі спеціальності 122 - Комп'ютерних наук, освітньо-професійної програми «Інформатика»
здобувача групи ІН-92 Кияненка Дмитра Віталійовича

1. Тема роботи: «Інформаційна система симуляції польоту рою дронів»
затверджую наказом по СумДУ від _____
2. Термін здачі здобувачем кваліфікаційної роботи *до 09 червня 2023 року*
3. Вхідні дані до кваліфікаційної роботи _____
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Аналіз проблеми предметної області, постановка й формування завдань дослідження. 2) Огляд аналогічних проектів, що вже існують. 3) Розроблення інформаційної системи польоту. 4) Аналіз результатів.*
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)_____
6. Консультанти до проєкту (роботи), із значенням розділів проєкту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «____» _____ 20 ___ р.

Завдання прийняв до виконання _____ Керівник _____
(підпис) (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Аналіз проблеми предметної області, постановка й формування завдань дослідження*		
2	*Огляд аналогічних проектів, що вже існують*		
3	*Розроблення інформаційної системи польоту*		
4	*Аналіз результатів*		
5	*Оформлення пояснювальної записки до кваліфікаційної роботи*		

Здобувач вищої освіти _____ Керівник _____
(підпис) (підпис)

АНОТАЦІЯ

Записка: 41 стр., 32 рис., 5 додатків, 17 використаних джерел.

Обґрунтування актуальності теми роботи – Тему кваліфікаційної роботи можна вважати доволі актуальною, так як на даний момент часу існує не так багато саме симуляцій польоту рою дронів, проте існує доволі багато симуляцій для управління одного дрона для того аби отримати навички його пілотування.

Об’єкт дослідження – політ рою дронів у симульованому середовищі.

Мета роботи – розробка інформаційної системи симуляції польоту рою дронів.

Методи дослідження – створення відповідної зони симуляції та створення в ній перешкод та задання початкових та кінцевих точок яких дрони повинні досягнути.

Результати – розроблено інформаційну систему, яка створює відповідне середовище для симуляції та при переміщенні дронів виводить всі їх дані, які потім заносяться до логів, також до окремих логів заносяться стани самого середовища

ІНФОРМАЦІЙНА СИСТЕМА, СИМУЛЯЦІЯ, ДРОНИ. РІЙ ДРОНІВ, C++, ООП

ЗМІСТ

ВСТУП

Актуальність. Тему кваліфікаційної роботи можна вважати доволі актуальною, так як на даний момент часу існує не так багато саме симуляцій польоту рою дронів, проте існує доволі багато симуляцій для управління одного дрона для того аби отримати навички його пілотування.

Об’єкт дослідження. Політ рою дронів у симульованому середовищі.

Предмет дослідження. Процесс польоту при заданих початкових та кінцевих точок.

Гіпотеза. Візуалізацію статистичних даних та керування ними можна досягнути шляхом застосування інформаційної технології, що реалізує сервіс для розміщення додатків які виконують ці функції.

Наукова новизна. Описане у даній роботі програмне рішення дозволить зрозуміти на доволі базовому рівні за якою логікою дрони здійснюють переміщення та отримати доволі базову візуалізацію цього процесу.

Структура. Дана робота складається зі вступу, аналітичного огляду, постановки задачі, вибору методу розв’язання поставленої задачі, опису програмного забезпечення інформаційної системи, висновків, списку використаних джерел та додатків.

1 АНАЛІТИЧНИЙ ОГЛЯД

1.1 Аналіз аналогічних проектів

При початковому пошуку аналогів було виявлено, що саме для симуляції польоту рою дронів не має відповідних аналогів, проте існує доволі багато таких які дозволяють повести саме симуляцію польоту одного дрона для того аби отримати деякі навички керування ним.

Також варто зазначити що хоча є і відповідні симуляторе, проте також існує можливість використати симуляції для польотів літаків, але досвід такого керування буде значно відрізнятися від спеціалізованих проектів які предзначені для цього.

Як вже зазначалося, симуляції дронів загалом використовують для того аби користувач отримав базові навички керування ними. Серед таких симуляторів можна зазначити такі як DJI Flight simulator[1], DJI Virtual flight[2], Zephyr Drone Simulator[3], так як вони є доволі популярними серед користувачів.

1.2 Огляд первинного коду симуляції

В основі цієї роботи використовувалась курсова робота з 2 курсу з темою «Симуляція польоту дронів». В ній потрібно було створити програму яка автоматично проведе дронів від початку до кінця.

Далі будуть наведені скріншоти виконання початкового коду програми яку було створено для курсової роботи. (рисунок 1.2.1 – 1.2.4)

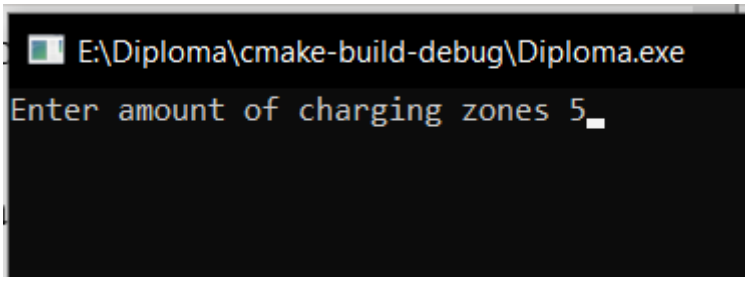

Рисунок 1.2.1 - Початкове вікно яке прохає ввести кількість зон для підзарядки

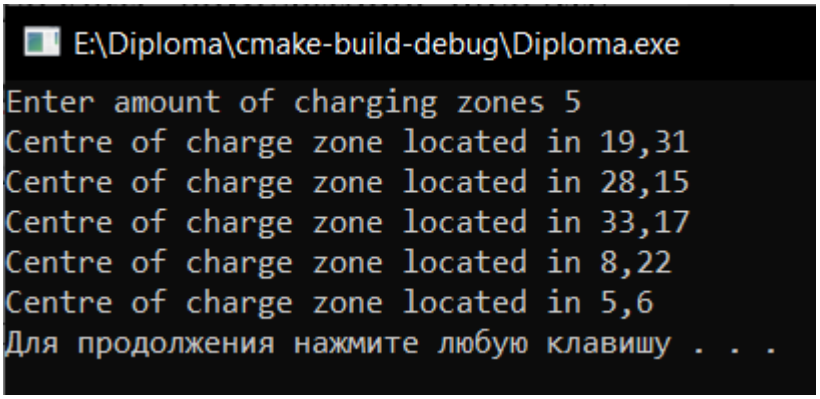

Рисунок 1.2.2 - Вікно після того як підтвердили кількість зон для зарядки

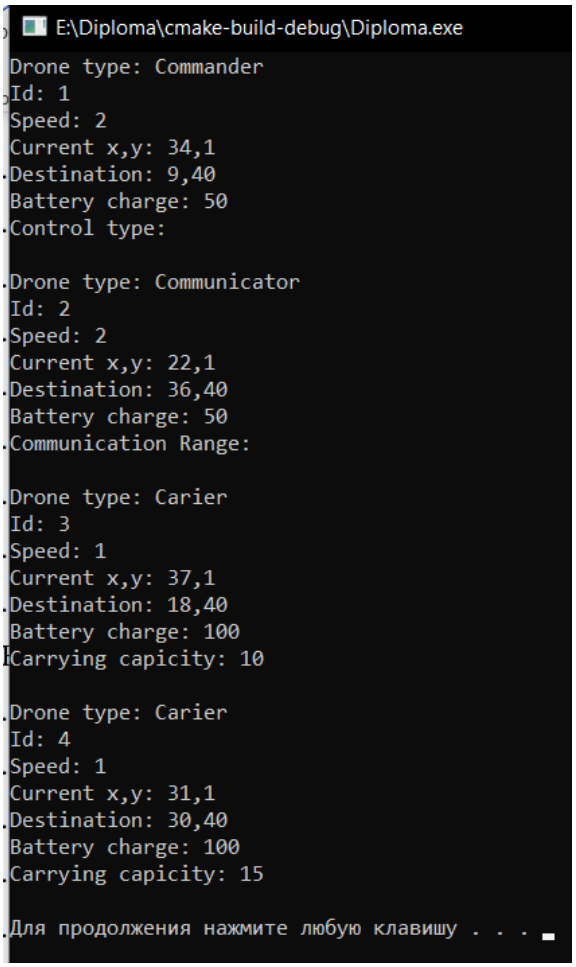

Рисунок 1.2.3 - Початкові позиції дронів та їх заряд

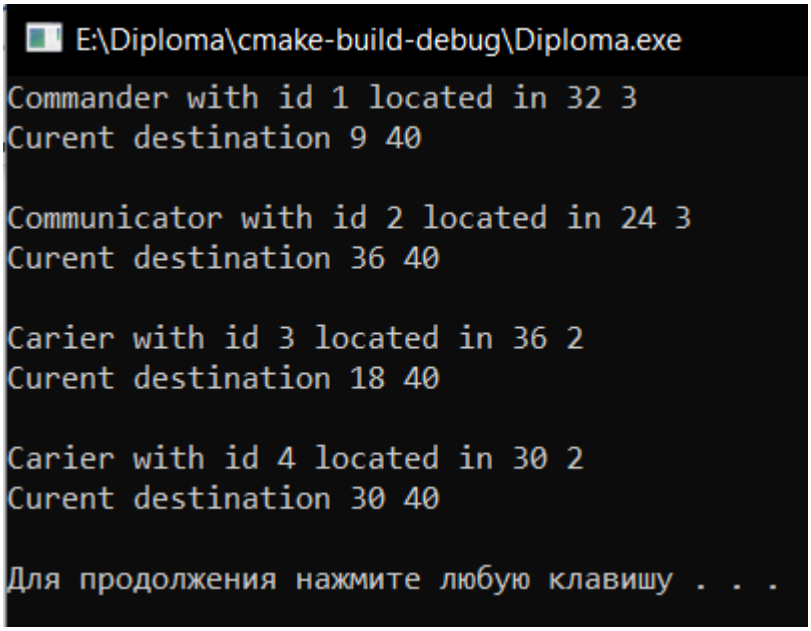

Рисунок 1.2.4 - Переміщення та вивід остаточної точки призначення

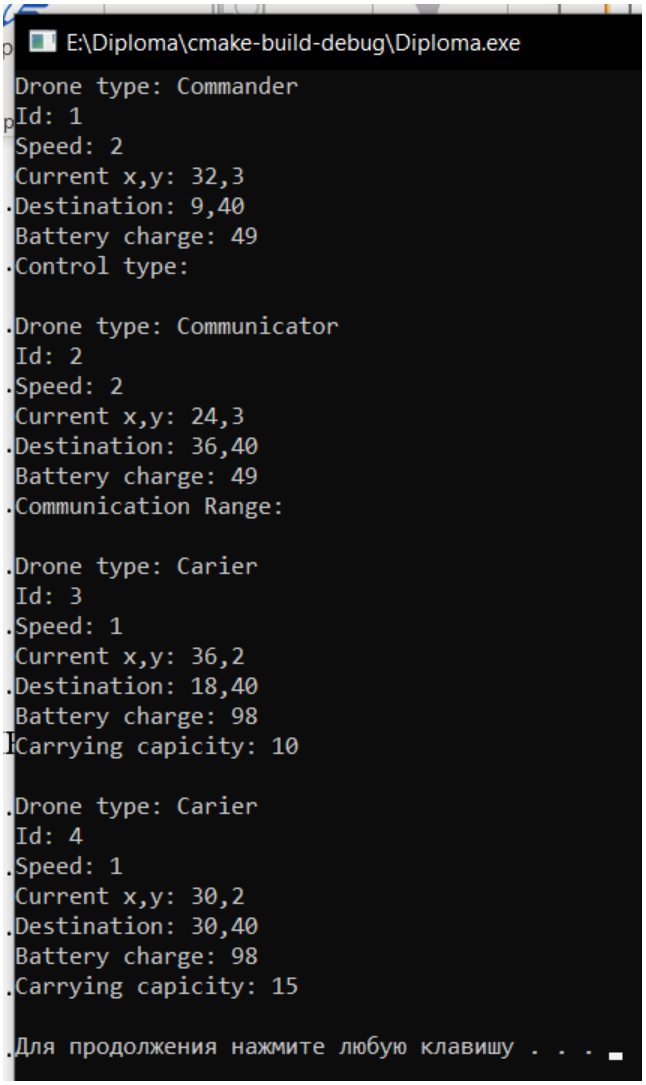

Рисунок 1.2.5 - Результат після переміщення

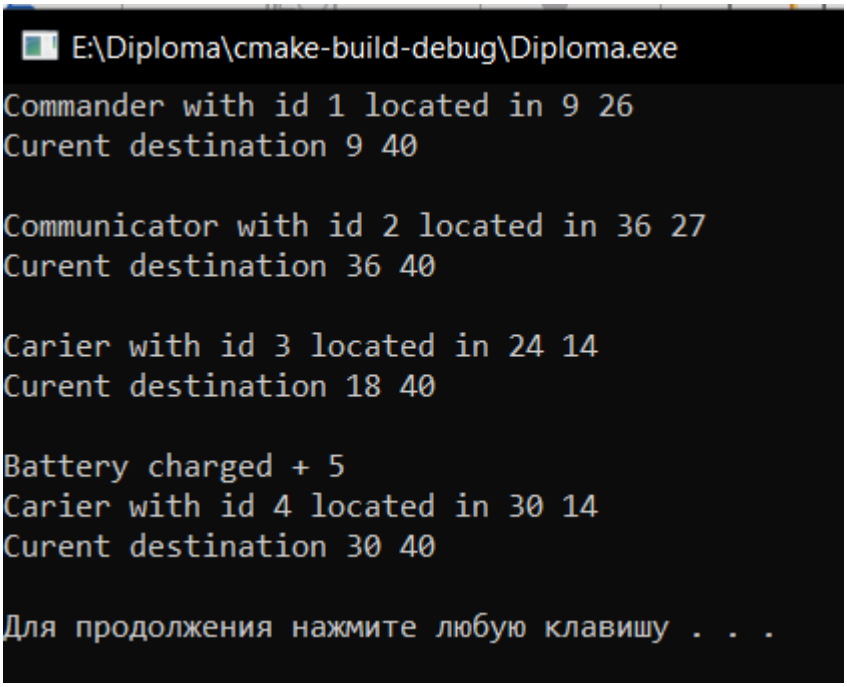

Рисунок 1.2.6 - Повідомлення про те що дрон потрапив на зону зарядки

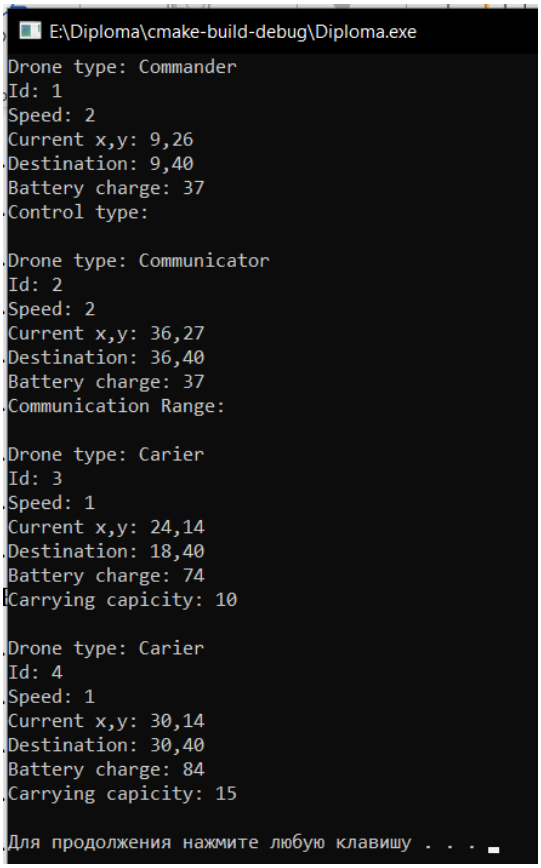

Рисунок 1.2.7 - Новий заряд батареї після потрапляння на зону підзарядки

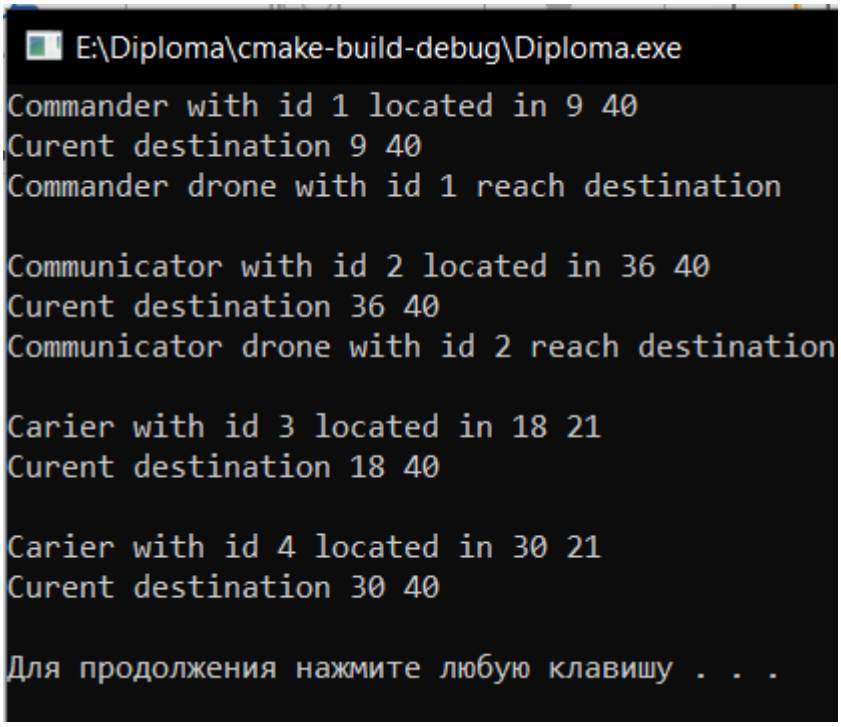

Рисунок 1.2.8 - Повідомлення про прибуття дронів у кінцеву точку

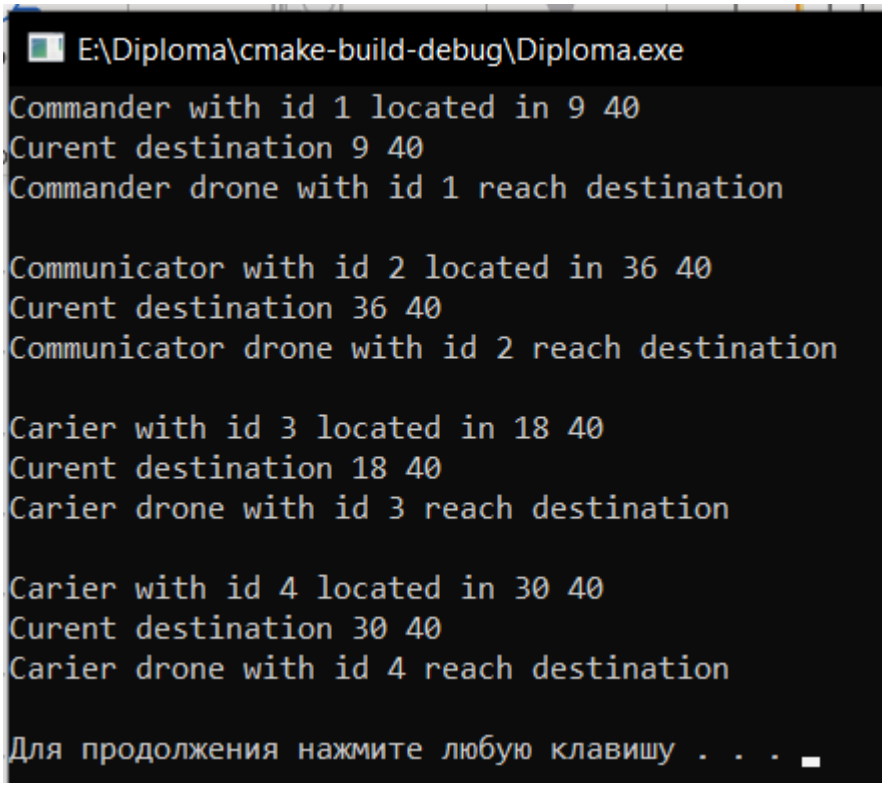

Рисунок 1.2.9 - Прибуття всіх дронів на кінцеву позицію

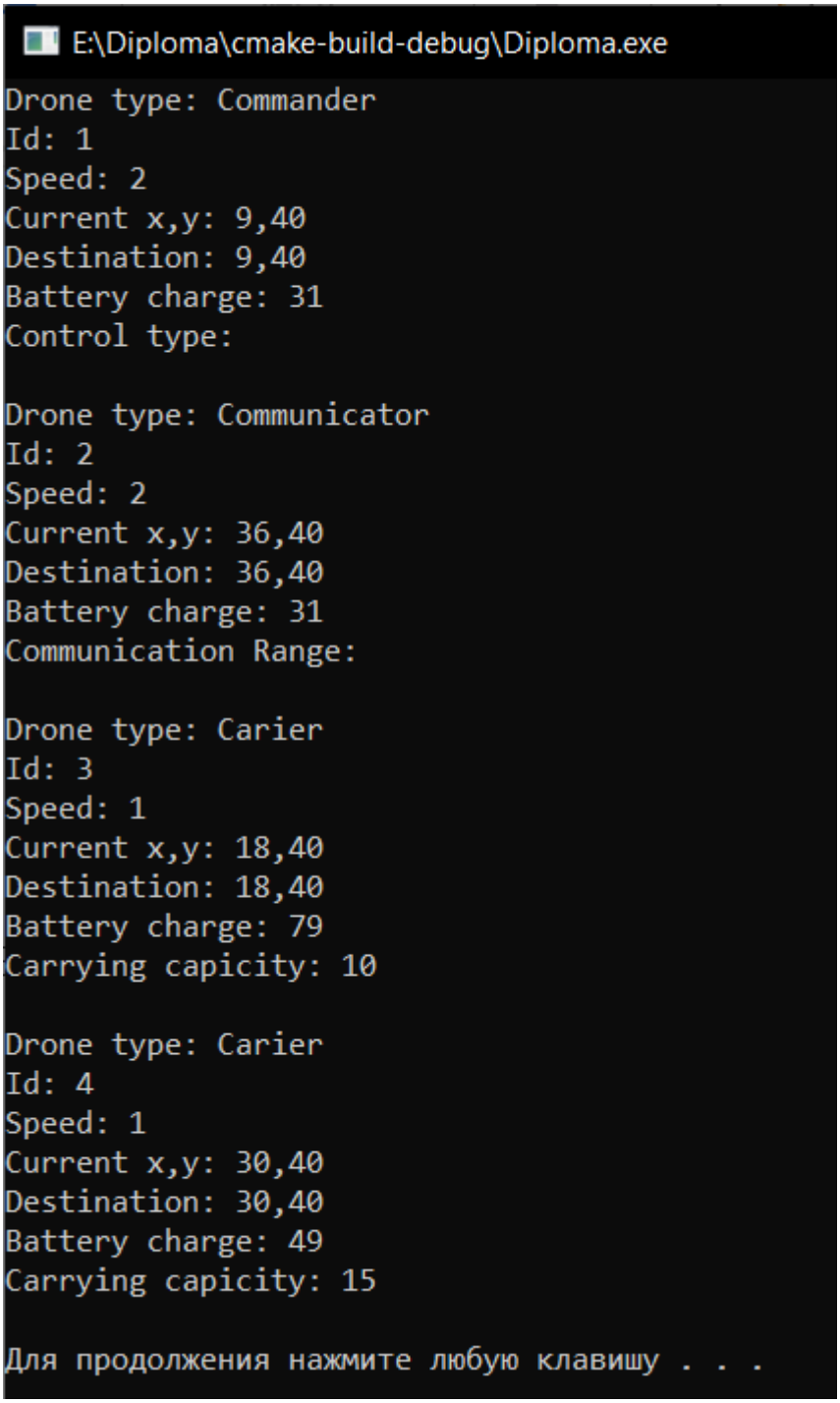

Рисунок 1.2.10 - Кінцевий результат після досягнення всіма дронами кінцевих

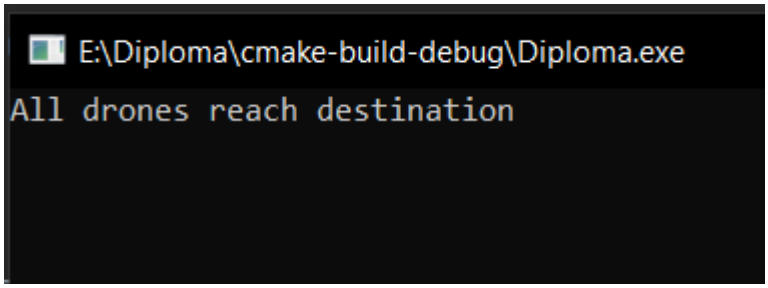

Рисунок 1.2.11 - Остаточне повідомлення підтвердження прибуття дронів

1.3 Постановка задачі

Метою роботи є створення симуляції польоту рою дронів у заданій області

Для досягнення поставленої мети необхідно вирішити наступні задачі:

1) Обрати та використати іншу мову програмування або залишити її тією самою;
2) Переписати початковий код до більш вищого стандарту;
3) Перевірити новий код на його відповідність до поставлених задач;
4) Завершити останню версію коду та перевірити її на наявність помилок

2 ВИБІР МЕТОДУ РОЗВ'ЯЗАННЯ ЗАДАЧІ

2.1 Вибір мови програмування

Так як вибір мови програмування може залежати від багатьох факторів, було вирішено обрати одну з найбільш популярних та ту з якою є більш-менш досвід праці.

За даними дослідження проведеними онлайн порталом DOU[4], опитування 9060 респондентів було отримано ось такий список. (рис. 2.1.1)

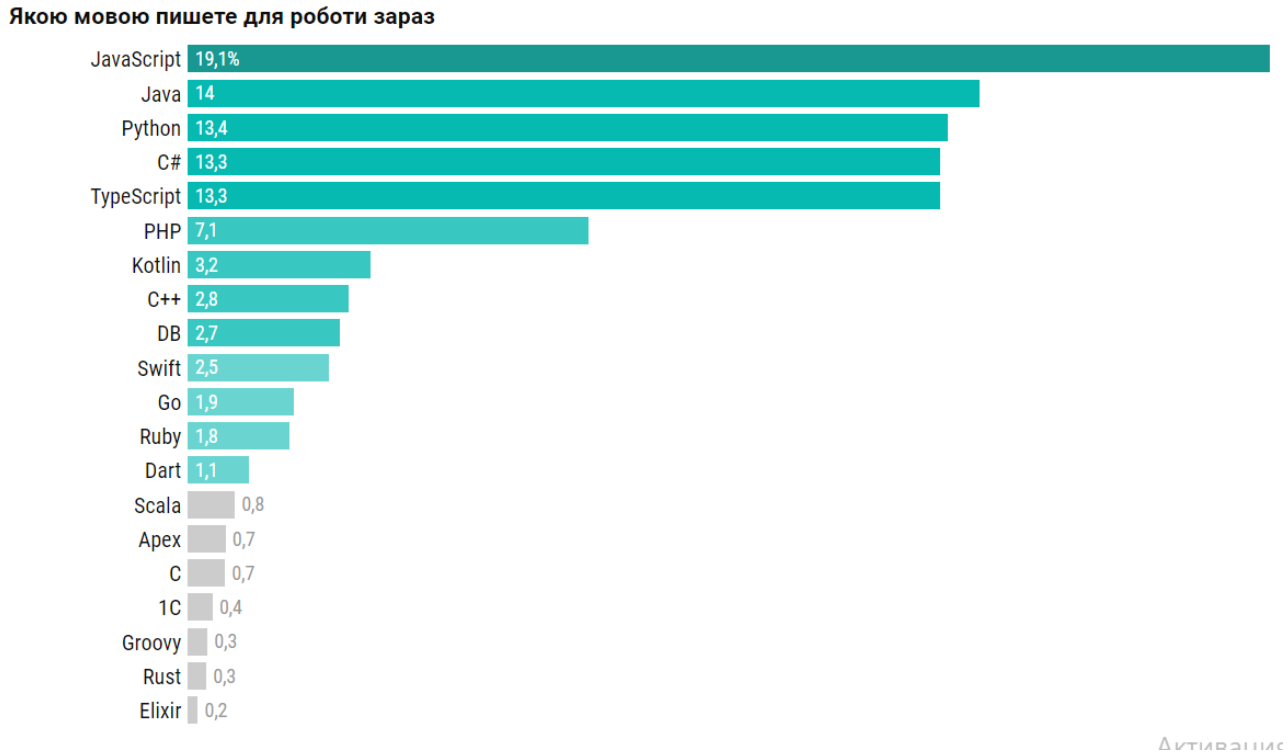

Рисунок 2.1.1 - Результат опитування респондентів

Одразу помітно що такі мови як JavaScript, Java, Python, C# є доволі популярними серед сучасних тенденцій у світі програмування. Кожна з них має свої переваги та недоліки, які впливають на їхню популярність.

Проте мовою програмування було обрано C++ оскільки вона є також доволі відомою та є достатньо швидкою та ефективною. Насамперед обрання мови було зумовлене тим що початковий код також був написаний на цій же мові, що в свою чергу дозволяє достатньо швидко перетворити цей код під потрібні стандарти, що в свою чергу не змушуватиме переписувати його заново

якби було обрано іншу мову програмування. Також важливо зазначити що у C++ є доволі багато бібліотек які можуть допомогти в подальшій розробці інформаційної системи.

2.2 Середовище розробки

Обрання середовища розробки може залежати від багатьох факторів. Починаючи від мови програмування, типу проекту, закінчуючи зручність інтерфейсу та підтримкою різних інструментів таких як Git.

Серед різних IDE було під увагу потрапили CLion[5], Visual Studio[6] та Notepad++[7]. Кожне з цих середовищ має свої власні переваги та недоліки які наведені у таблиці 1.1.

Таблиця 1.1 – Порівняння різних IDE для розробки

Назва середовища	**Переваги**	**Недоліки**	**Ознайомленість**
CLion	Була створена спеціально для мови програмування C++. Має для неї доволі розширену підтримку. Має доволі гарні інструменти для рефоктору коду, та аналізу помилок. Існує система керування версіями, такими як Git.	Є платною, проте існує безкоштовна версія Community Edition. Є доволі ресурсомістка, що в свою чергу може сказатися на роботі на старих або слабких комп’ютерах.	Присутня ознайомленість з іншим продуктом від цього самого розробника – JetBrains. Тобто всі основні елементи управління уже знайомі.
Visual Studio	Присутній доволі значний набір інструментів для розробки на доволі	Доволі ресурсомістка що в свою чергу може сповільнювати	Присутня ознайомленість, оскільки на початкових курсах

		роботу менш	основні проекти

Продовження табл. 1.1

	різних мовах програмування, таких як C++, C#, Python та інші. Присутня інтегрована відладка яка має доволі багато можливостей. Присутній доволі широкий вибір плагінів та розширень для додаткового функціоналу.	потужних пристроїв. Деякі функції та разширення доступні лише в платних версіях. Існує підтримка лише для Windows, що істотно обмежує написання проектів для будь-якиї інших платформ.	були написані саме в цьому середовищі.
Notepad++	Середовище яке є використовує доволі мало системних ресурсів. Присутня підтримка для багатьох мов програмування. Безкоштовна та має відкритий код, також має доволі активну спільноту користувачів	Обмежена функціональність в порівнянні з двома попередніми IDE. Відсутні вбудовані інструменти для відладки програм.	Ознайомленість майже відсутня.

Ознайомившись з перевагами та недоліками було прийнято рішення використовувати середовище розробки CLion, оскільки за останні пару років

ознайомленість з його аналогом для мови програмування Java була на доволі високому рівні.

2.3 Вибір системи пошуку шляху

Так як нам потрібно аби дрони долетіли до кінцевої точки, вони повинні знайти шлях за яким вони будуть туди добиратися, тобто їм потрібна логіка поведінки якою вони будуть керуватися для досягнення мети.

Для того аби досягнути кінцевої точки існує багато алгоритмів пошуку шляху, тому було розглянуто наступні: A*[8], DFS(Deep First Search)[9], BFS(Breadth First Search)[10], та початковий код для пошуку шляху (див. Додаток А). Перваги та недоліки наведені далі у таблиці 2.1.

Таблиця 2.1 – Порівняння алгоритмів пошуку шляху

Назва	Переваги	Недоліки
A*(A-star)	Доволі ефективний алгоритм пошуку, який комбінує найкращий перший шлях з використання евристичної функції. Присутня оптимальна властивість що означає, що знайдений шлях буде найкоротшим, якщо евристична функція буде нормальною	Присутнє обчислення еврестиної функції для кожного вузла, що в свою чергу веде до того що він є доволі ресурсномістким. Не гарантована найбільша продуктивність на великих графах
DFS(Deep First Search)	Доволі простий алгоритм котрий використовує глибинний пошук для пошуку шляху. Гарно працює на великих графах, або там де потрібно перевірити всі можливі	Не гарантовано знаходження найкоротшого шляху, оскільки може потрапити на гілку котра не веде до цілі. Не підходить для графі де кожному ребру присвоєна вага.

	шляхи.	
BFS(Breadth First Search)	Дозволяє знаходити найкоротший шлях в графах	Є доволі ресурсномістким. Не ефективний на графах з

Продовження табл. 2.1

	з однаковими вагами ребр. Можна використовувати для знаходження всіх можливих шляхів у відповідному діапазоні від стартового вузла	великою кількістю вузлів.

Серед цих чотирьох три з них окрім початкового коду використовують для пошуку шляху графи. Тобто вони перетворюють початкову зону в граф, в якому потім і проводять обчислення для найкращого шляху. Проте при їх аналізі виявилось що майже неможливо зробити так аби після того як знайшовся шлях, аби по ньому рухалися з відповідною швидкістю, рух відбувався лише на 1 «вузол», що не відповідало потрібним умовам руху, адже кожен з дронів мав можливість мати іншу швидкість, що в свою чергу змушувало використовувати інший метод пошуку шляху та його досягнення.

3 ІНФОРМАЦІЙНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

3.1 Поняття зони польотів

Так як в початковій версії дрони літали в двовимірному масиві, то було вирішено що це так і залишиться. Також було прийнято рішення про те аби переміщення дронів відображалось графічно. На рисунку 3.1.1 показано спрощений графік як виглядає масив зони польоту.

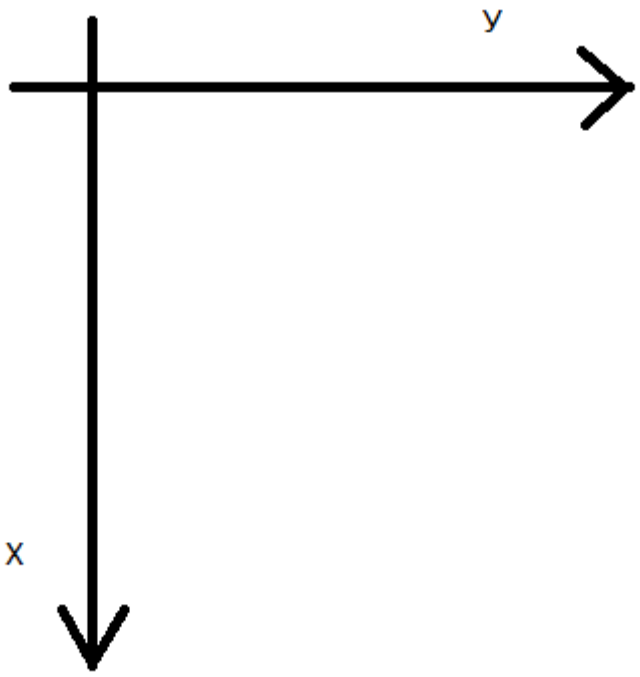

Рисунок 3.1.1 - Схематичний рисунок для представлення двовимірного масиву

Де X – це номер рядку, а У – номер стовпчика у двовимірному масиві, що відповідає загальному визначенню двовимірного масиву (рис. 3.1.2 [11])

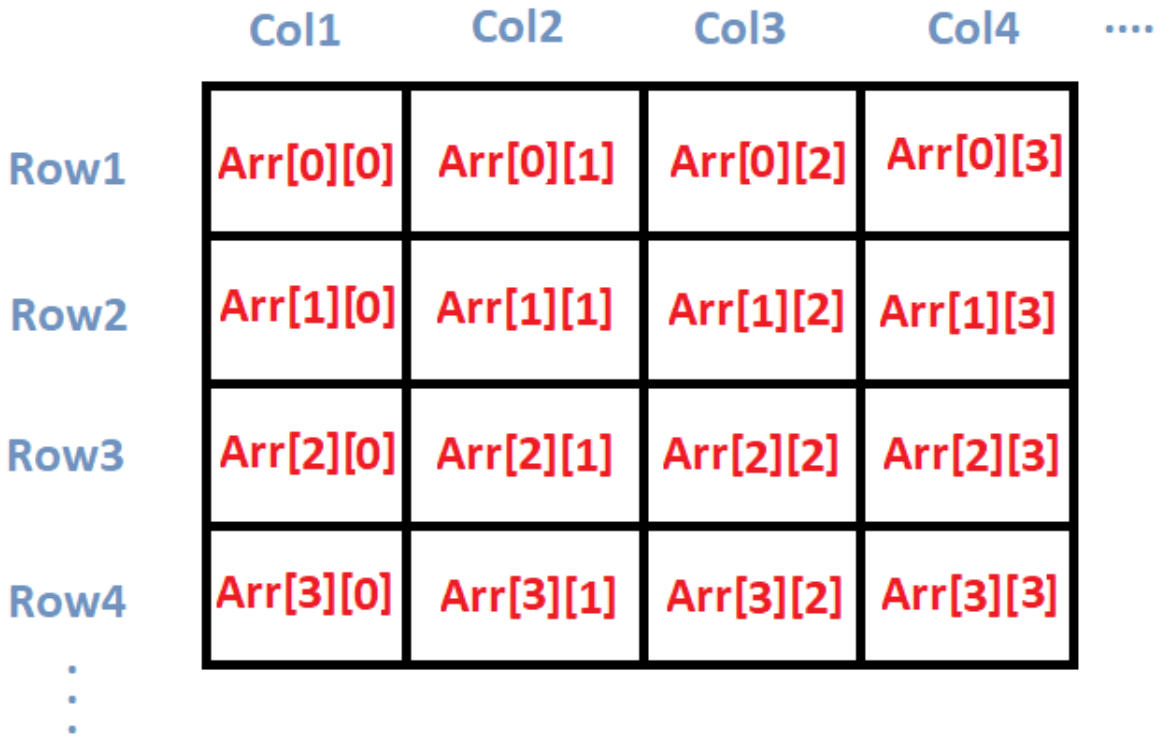

Рисунок 3.1.2 – Стандартний вигляд двовимірного масиву

Також для того аби графічне відображення було біль приємним оку було обрано варіант в якому дрони рухаються з найбільшого ряду до найменшого. (рисунок 3.1.3)

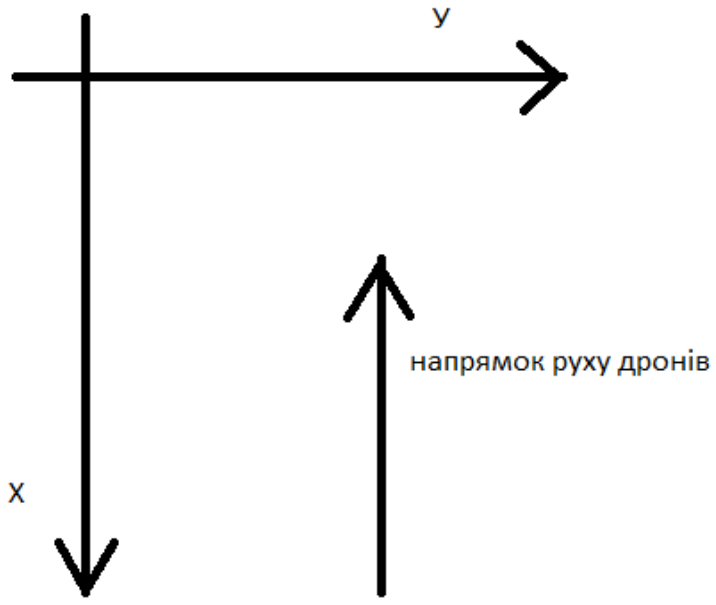

Рисунок 3.1.3 - Спрощене зображення напрямку руху дронів

3.2 Опис програмної реалізації

Спочатку було створено сторінку запуску в котрій присутні назва та псевдо анімація (рис. 3.2.1) (код див. Додаток Б).

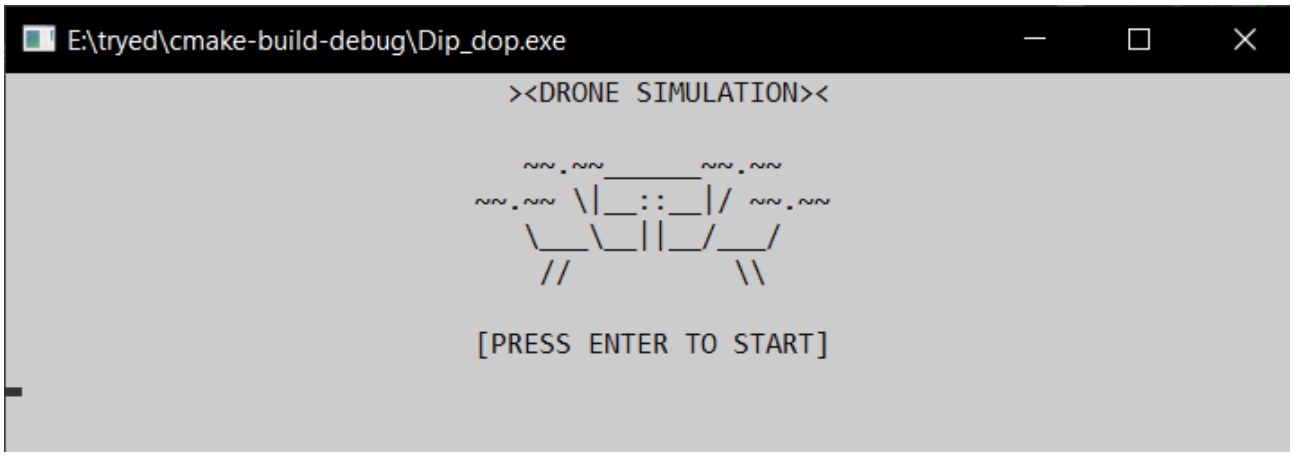

Рисунок 3.2.1 - Інтро яке бачить користувач при запуску програми

Після того як користувач натисне клавішу ENTER його перанаправить до меню в якому йому запропонують на вибір один з двох варіантів, а також йому повідомлять що для того аби закрити програму можна натиснути клавішу ESCAPE(рис. 3.2.2).

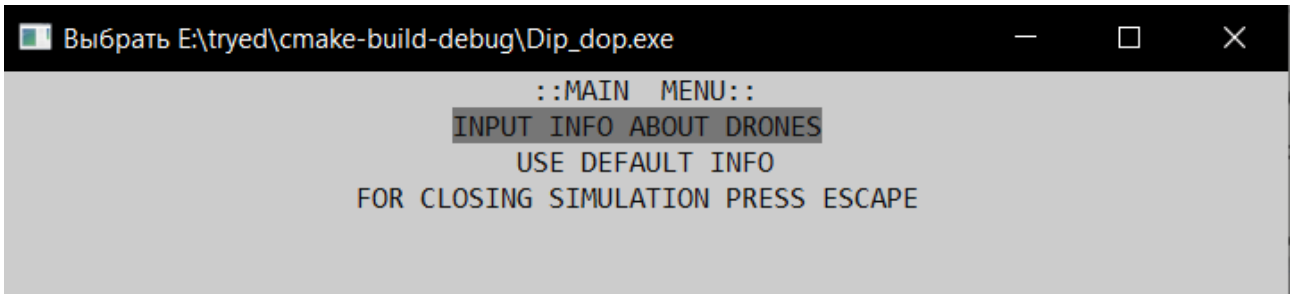

Рисунок 3.2.2 - Меню вибору користувача

Важливо зазначити що можливість закрити програму за допомогою клавіші ESCAPE присутня тільки до моменту запуску самої симуляції, тобто після запуску польоту закрити програму можливо буде лише за допомогою натискання на хрестик у правому куті вікна програми.

Якщо користувач вирішить обрати стандартні дані, то йому потрібно буде натиснути на клавіатурі клавішу вниз аби обрати цю функцію, після цього

програма перенесе його до наступного меню та повідомить про те що стандартні дані будуть використовуватися(рис. 3.2.3, рис. 3.2.4).

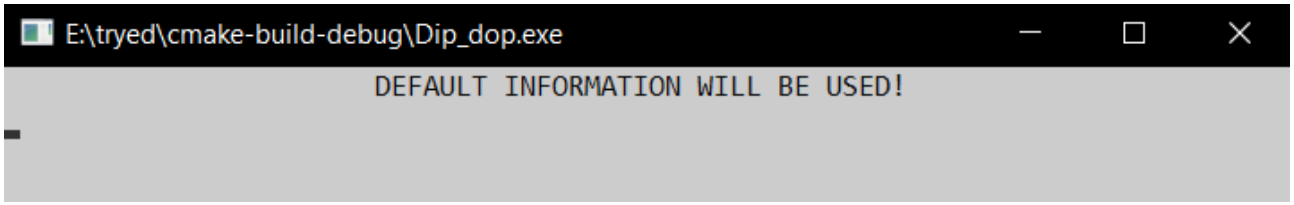

Рисунок 3.2.3 - Повідомлення про використання стандартних даних

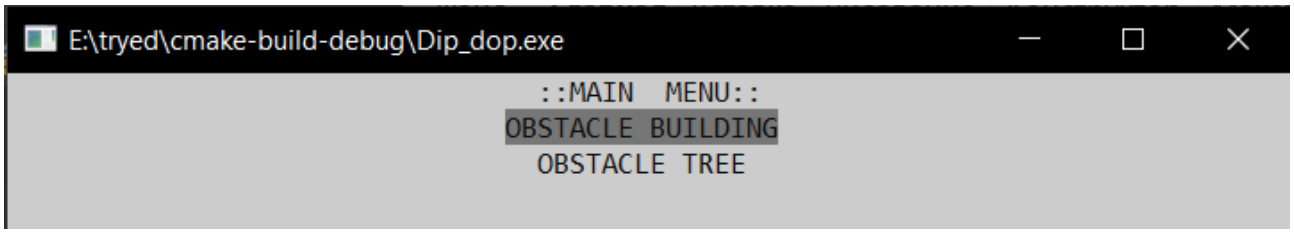

Рисунок 3.2.4 - Меню вибору перешкод

Якщо ж користувач забажає сам увести дані для дронів його перенаправить на відповідну сторінку(рис 3.2.5, рис 3.2.6).

Рисунок 3.2.5 - Початковий вигляд форми

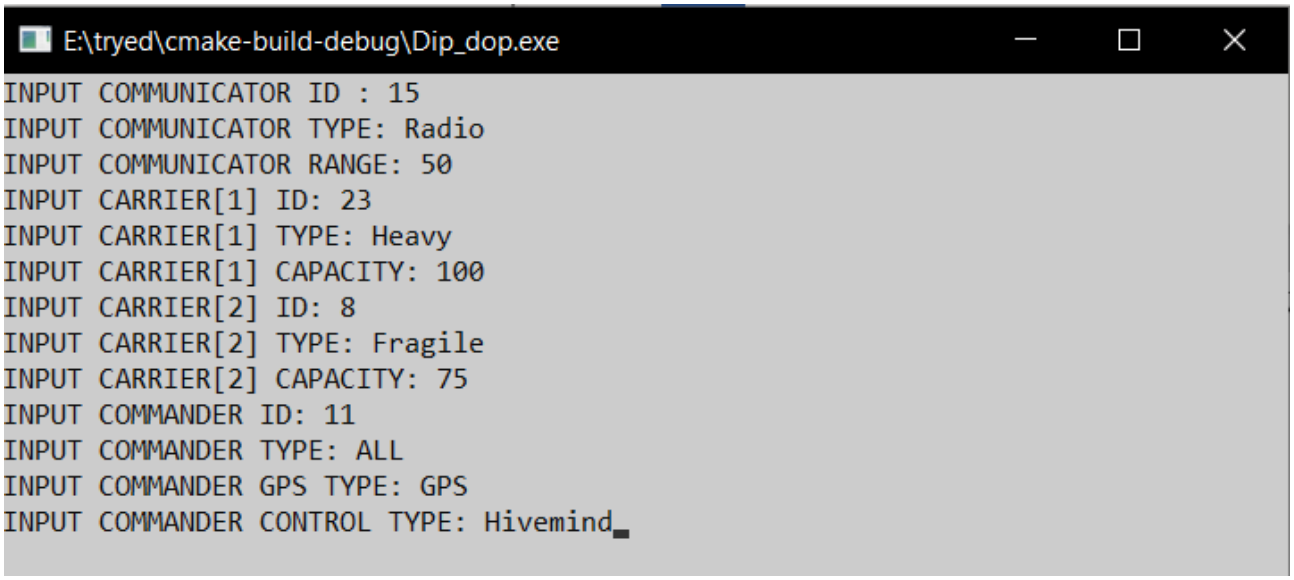

Рисунок 3.2.6 - Вигляд форми пілся її заповнення

Важливо зазначити що всі поля виводяться поступово і тільки після того як користувач ввів якісь дані. Також важливим є те що система не дозволить користувачу ввести однакові ID для дронів, а присвоїть йому наступний ID який не вказаний в основних даних, тобто 5.

В меню вибору перешкод(рис 3.2.4) представлено два варіанти, кожен з яких буде відрізнятися кількістю та графічним видом перешкоди. (для Building див Додаток В, для Tree див. Додаток Г)

Після вибору бажаного типу перешкод користувач буде перенаправлений на наступне і останнє меню(рис 3.2.7).

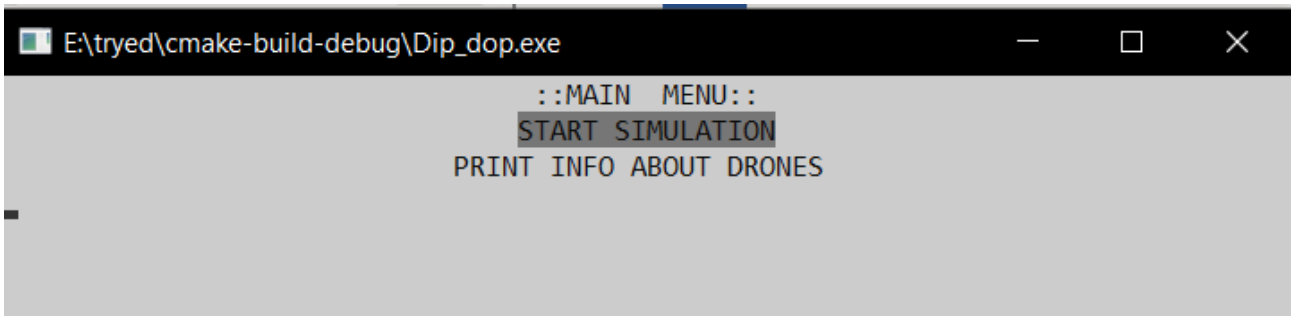

Рисунок 3.2.7 - Останнє меню перед запуском симуляції

Якщо користувач забажає переглянути всю інформацію яка відома про дронів то він повинен обрати за допомогою стрілочок другий пункт. Після того як він буде обраний, з’явиться вікно в якому будуть всі дані про дрони та дані про обраний тип перешкод(рис 3.2.8, рис 3.2.9).

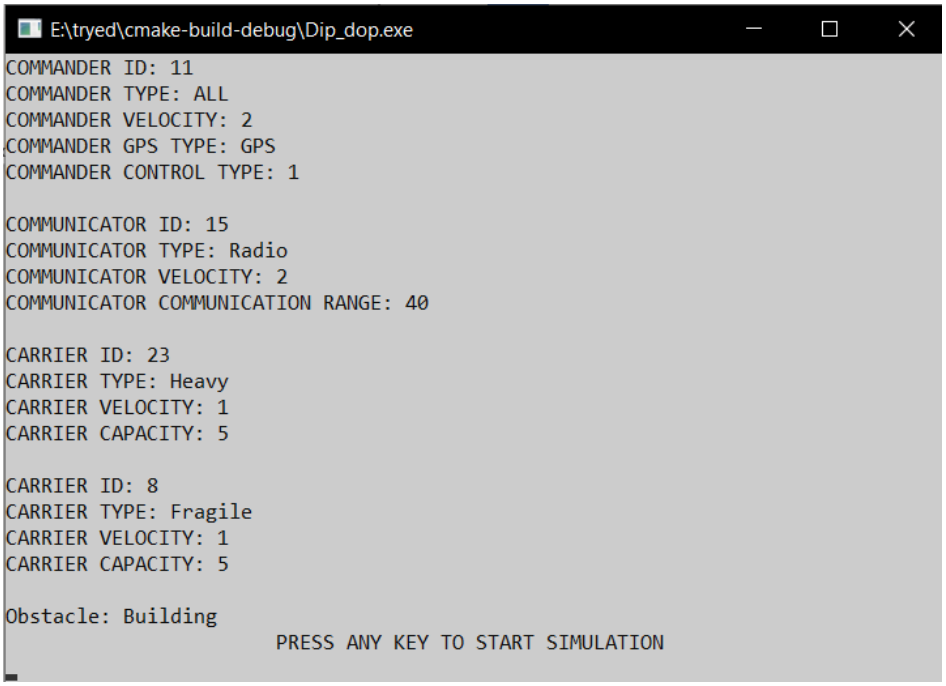

Рисунок 3.2.8 - Дані які вводив користувач

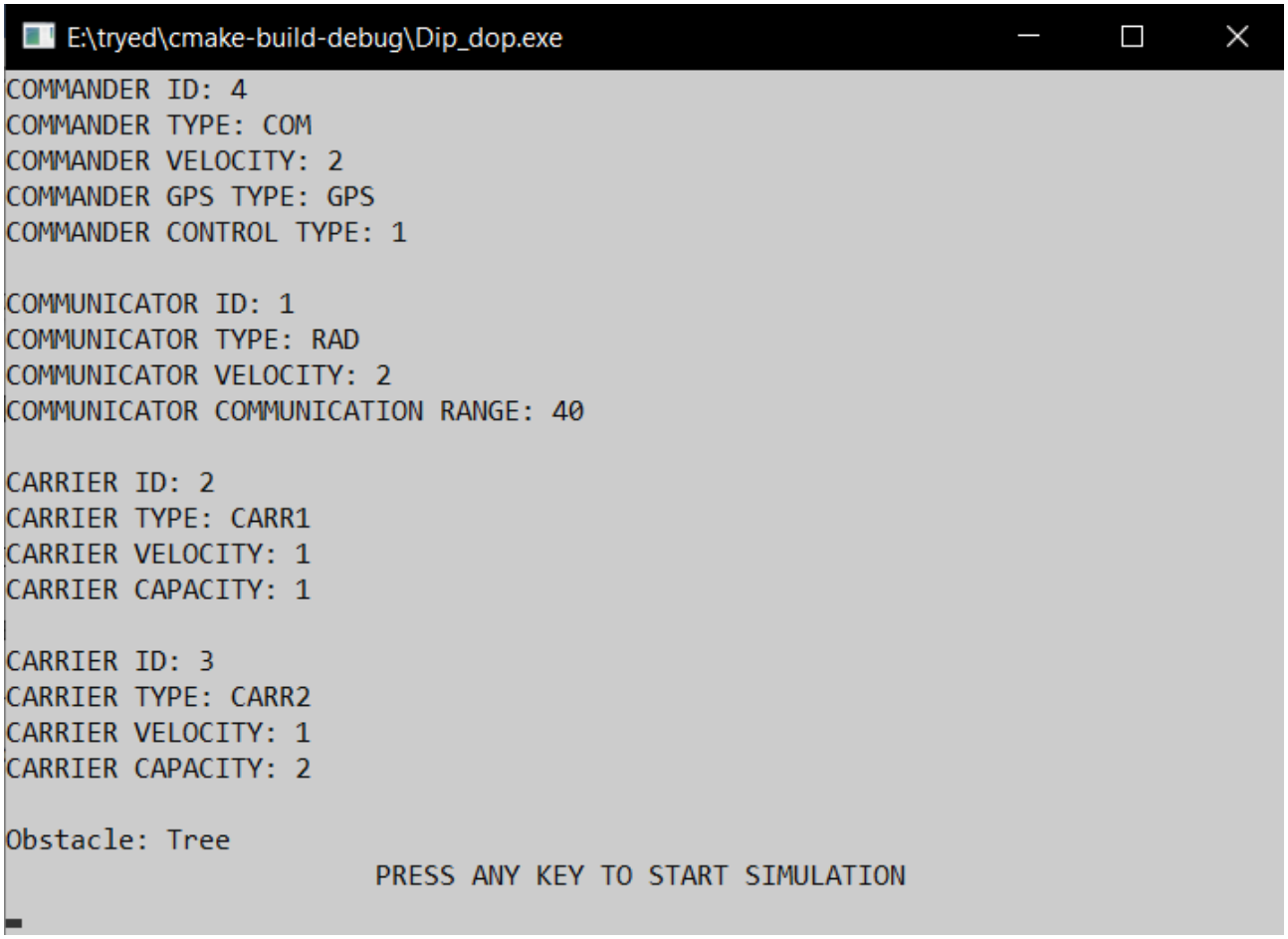

Рисунок 3.2.9 - Стандартні дані з вибраним типом перешкоди Tree

Після того як користувач переконався у правдивості даної інформації він повинен натиснути будь-яку клавішу для того аби запустити симуляцію, або ж

якщо він не захоче її перевіряти та обере одразу START SIMULATION його перенесу одразу на екран симуляції дронів яка одразу ж почнеться (рисунок 3.10).

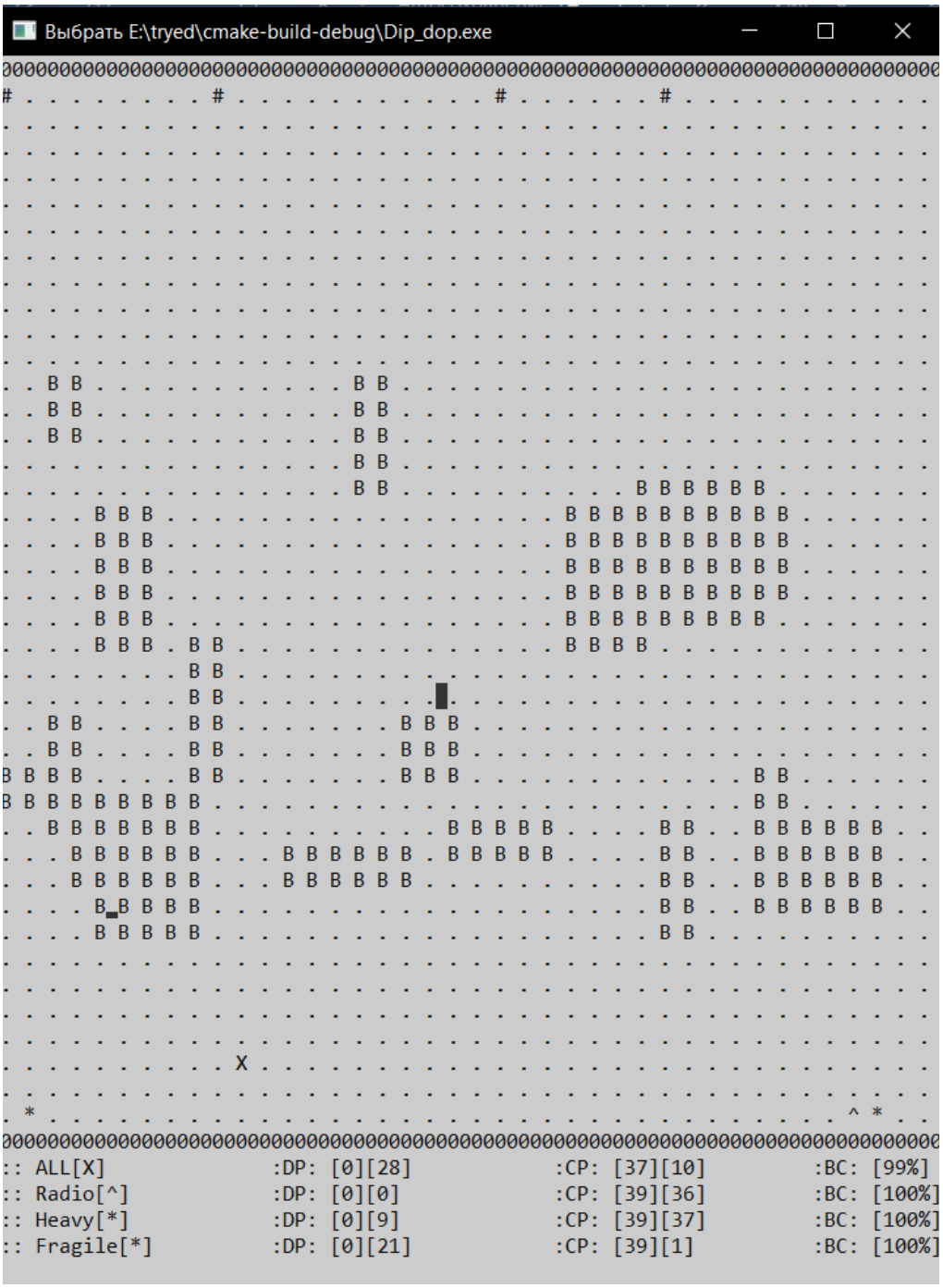

Рисунок 3.2.10 – Початок симуляції

Важливо зазначити що фігури які позначені як # це фінальна точка в яку потрібно прилетіти дронам, а фігури які позначені як B це перешкоди які відповідають типу Building , в той час, фігури які будуть позначені як T будуть

відповідно перешкодами типу Tree. Також важливим є той факт що при потраплянні на перешкоду дрон буде вважатися знищеним, що в свою чергу відображатиметься тим що він зникне з графічного відображення та його координати та заряд не будуть змінюватися (рис.3.2.11, рис 3.2.12).

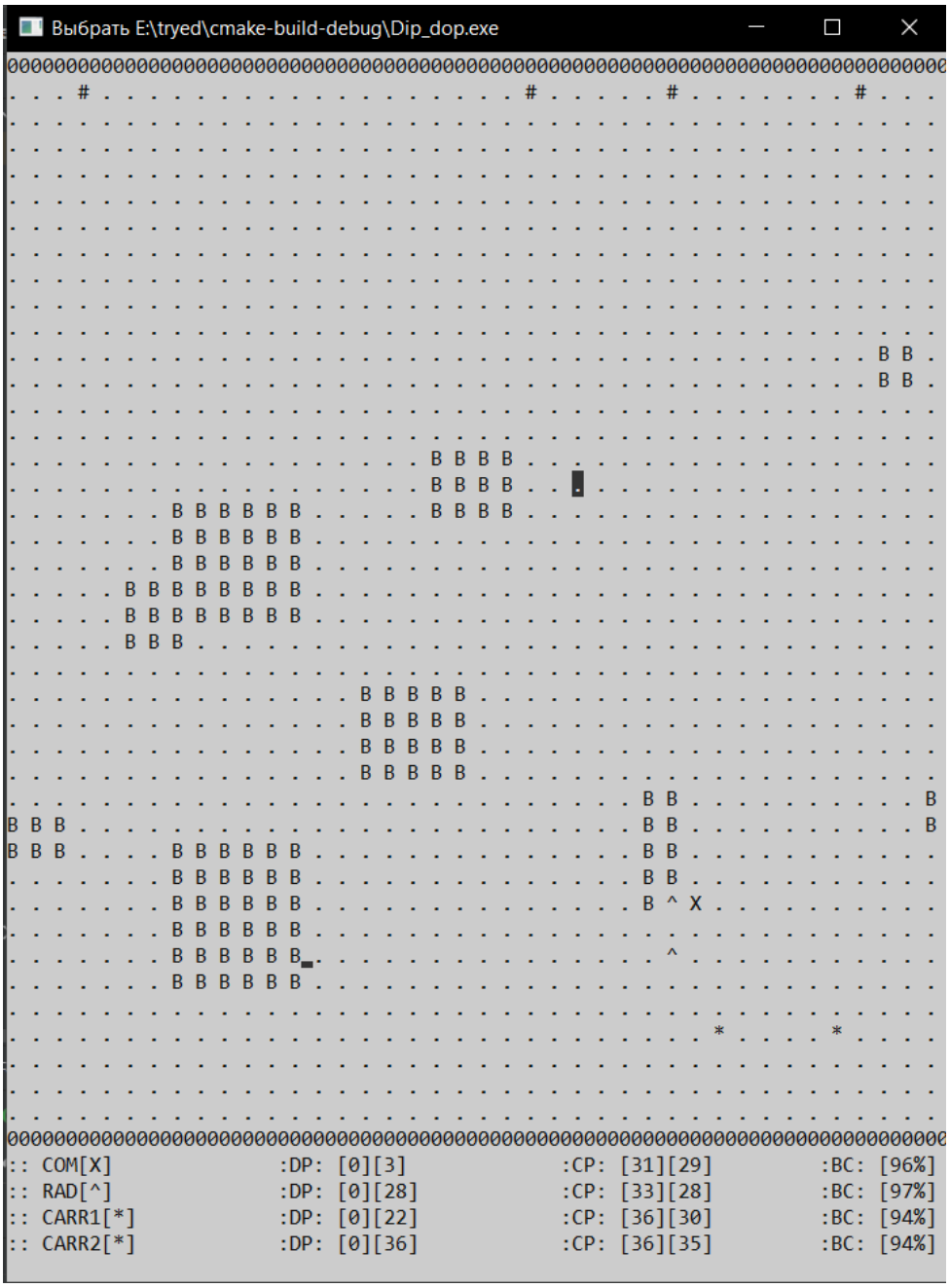

Рисунок 3.2.11 – Дрон потрапив на перешкоду

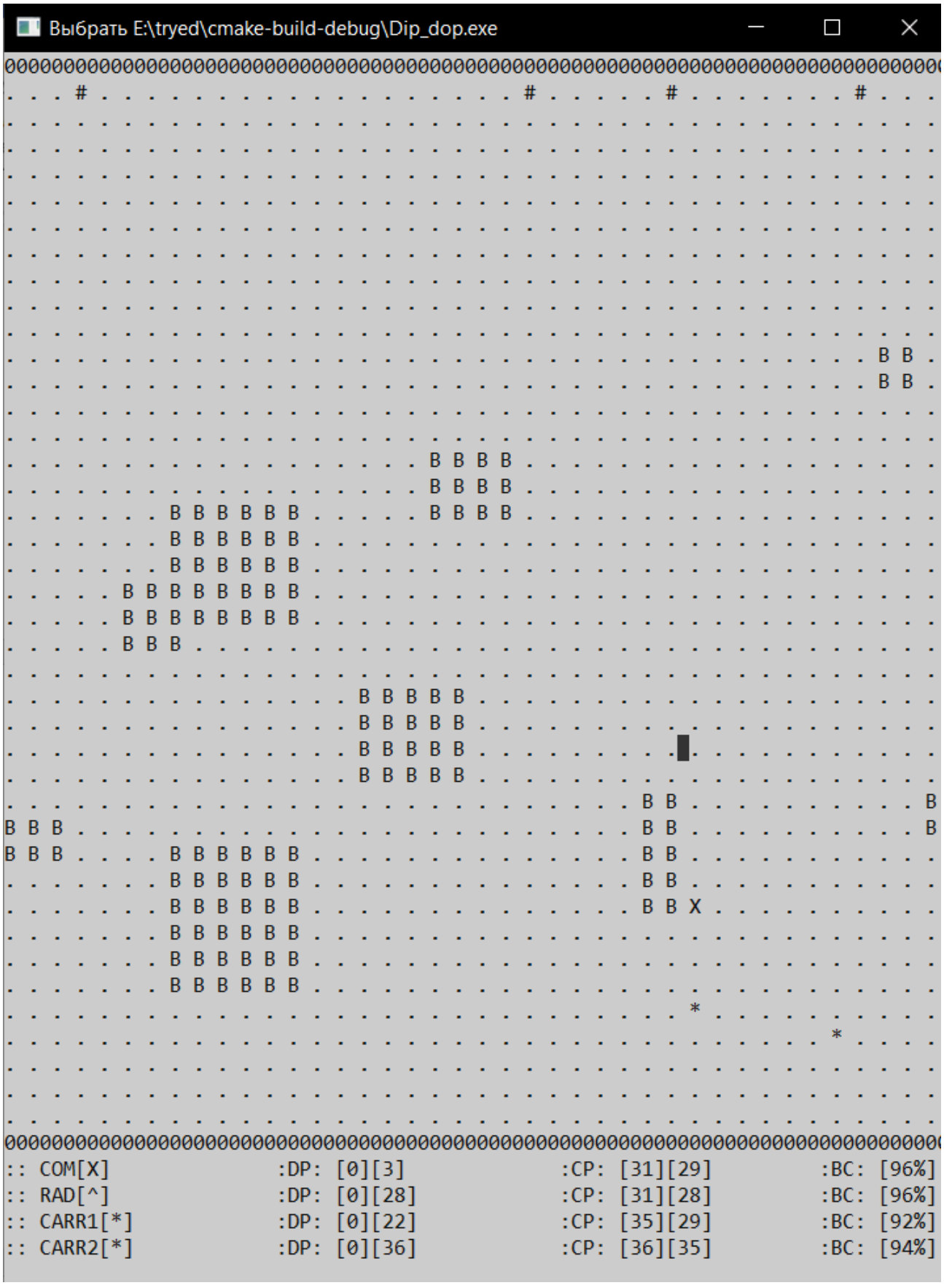

Рисунок 3.2.12 - Дрон вважається знищеним

Також як можна бачити на рисунку 3.2.10 та рисунках 3.2.11-12 місцеположення перешкод змінюється з кожним новим запуском, змінюються також і стартові позиції та кінцеві, тобто майже неможливо отримати дві однакові симуляції якщо звісно не змінювати код.

Такий вигляд має вікно коли усі дрони які залишились долетіли до свого місця призначення (рисунок 3.2.13).

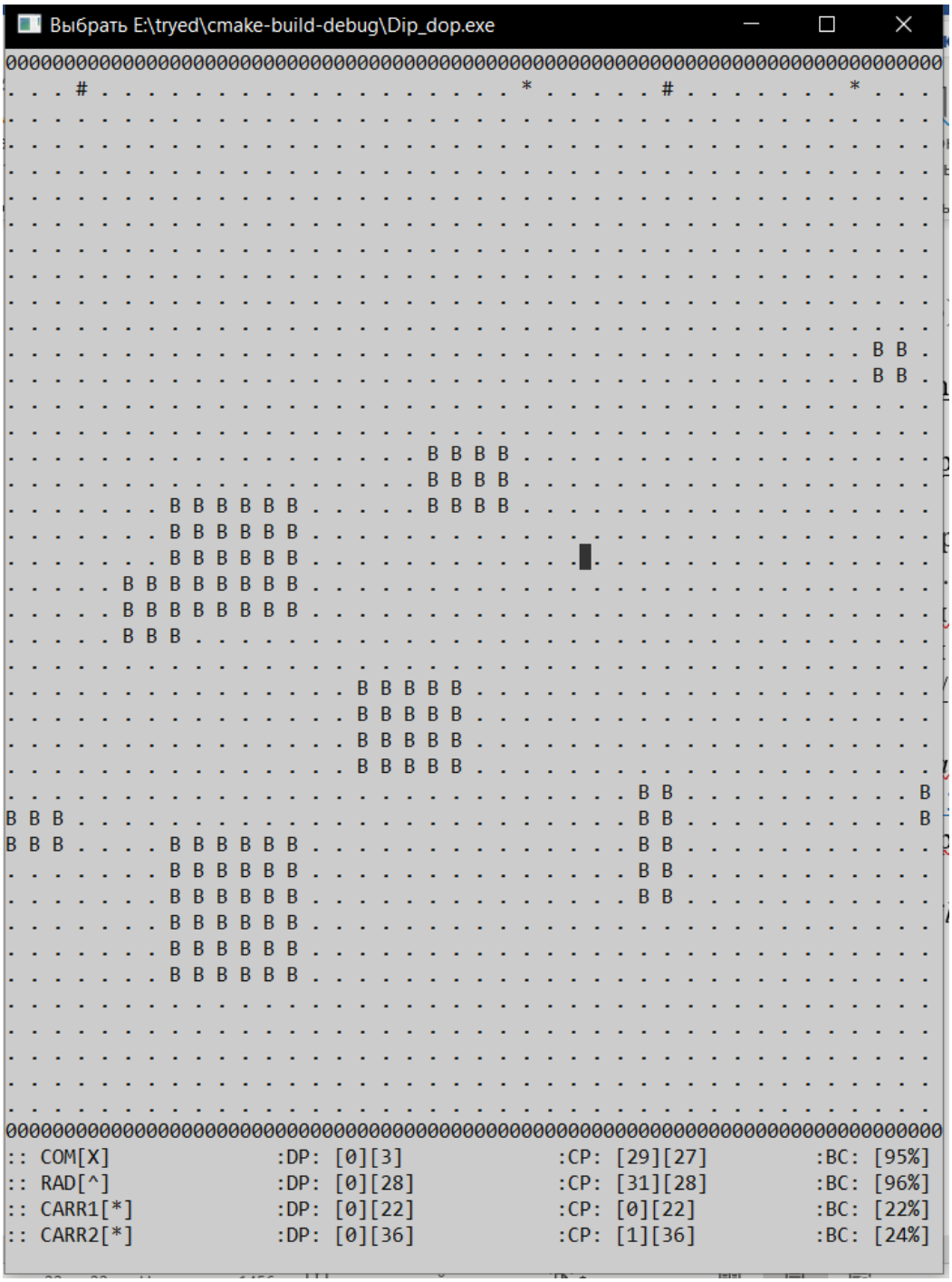

Рисунок 3.2.13 - Кінцевий вигляд симуляції

Після того як симуляція завершиться з’явиться повідомлення в якому буде написано SIMULATION IS OVER!, після цього програма автоматично завершить своє виконання.

Також при виконанні програма створює логи польоту та логи самої зони для польоту що дозволить перевірити перед наступним запуском дані котрі були в останній раз. (рис. 3.2.14-3.2.17)

Рисунок 3.2.14 - Загальний вигляд логу польоту

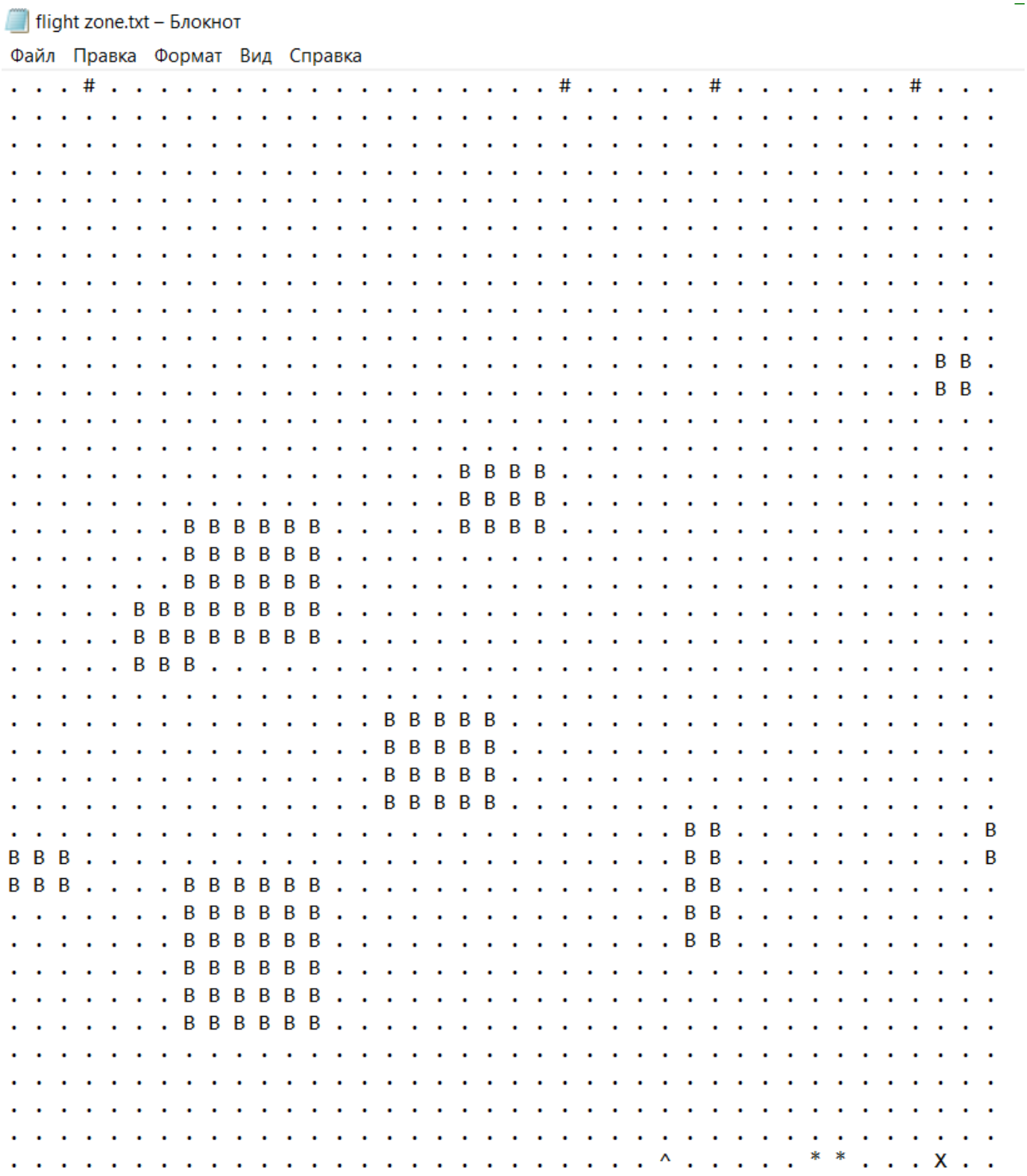

Рисунок 3.2.15 - Відповідний логу польотів лог зони симуляції

log.txt – Блокнот

Файл Правка Формат Вид Справка

```
COM[X]  DESTINATION: [0][14]      POSITION: [37][28]     BATTERY: [99%]
RAD[^]  DESTINATION: [0][21]      POSITION: [37][23]     BATTERY: [99%]
CARR1[*]        DESTINATION: [0][36]      POSITION: [38][32]     BATTERY: [98%]
CARR2[*]        DESTINATION: [0][32]      POSITION: [38][1]      BATTERY: [98%]



COM[X]  DESTINATION: [0][14]      POSITION: [35][26]     BATTERY: [98%]
RAD[^]  DESTINATION: [0][21]      POSITION: [35][21]     BATTERY: [98%]
CARR1[*]        DESTINATION: [0][36]      POSITION: [37][33]     BATTERY: [96%]
CARR2[*]        DESTINATION: [0][32]      POSITION: [37][2]      BATTERY: [96%]



COM[X]  DESTINATION: [0][14]      POSITION: [33][24]     BATTERY: [97%]
RAD[^]  DESTINATION: [0][21]      POSITION: [33][21]     BATTERY: [97%]
CARR1[*]        DESTINATION: [0][36]      POSITION: [36][34]     BATTERY: [94%]
CARR2[*]        DESTINATION: [0][32]      POSITION: [36][3]      BATTERY: [94%]



COM[X]  DESTINATION: [0][14]      POSITION: [31][22]     BATTERY: [96%]
RAD[^]  DESTINATION: [0][21]      POSITION: [31][21]     BATTERY: [96%]
CARR1[*]        DESTINATION: [0][36]      POSITION: [35][35]     BATTERY: [92%]
CARR2[*]        DESTINATION: [0][32]      POSITION: [35][4]      BATTERY: [92%]
```

Рисунок 3.2.16 - Лог іншого польоту

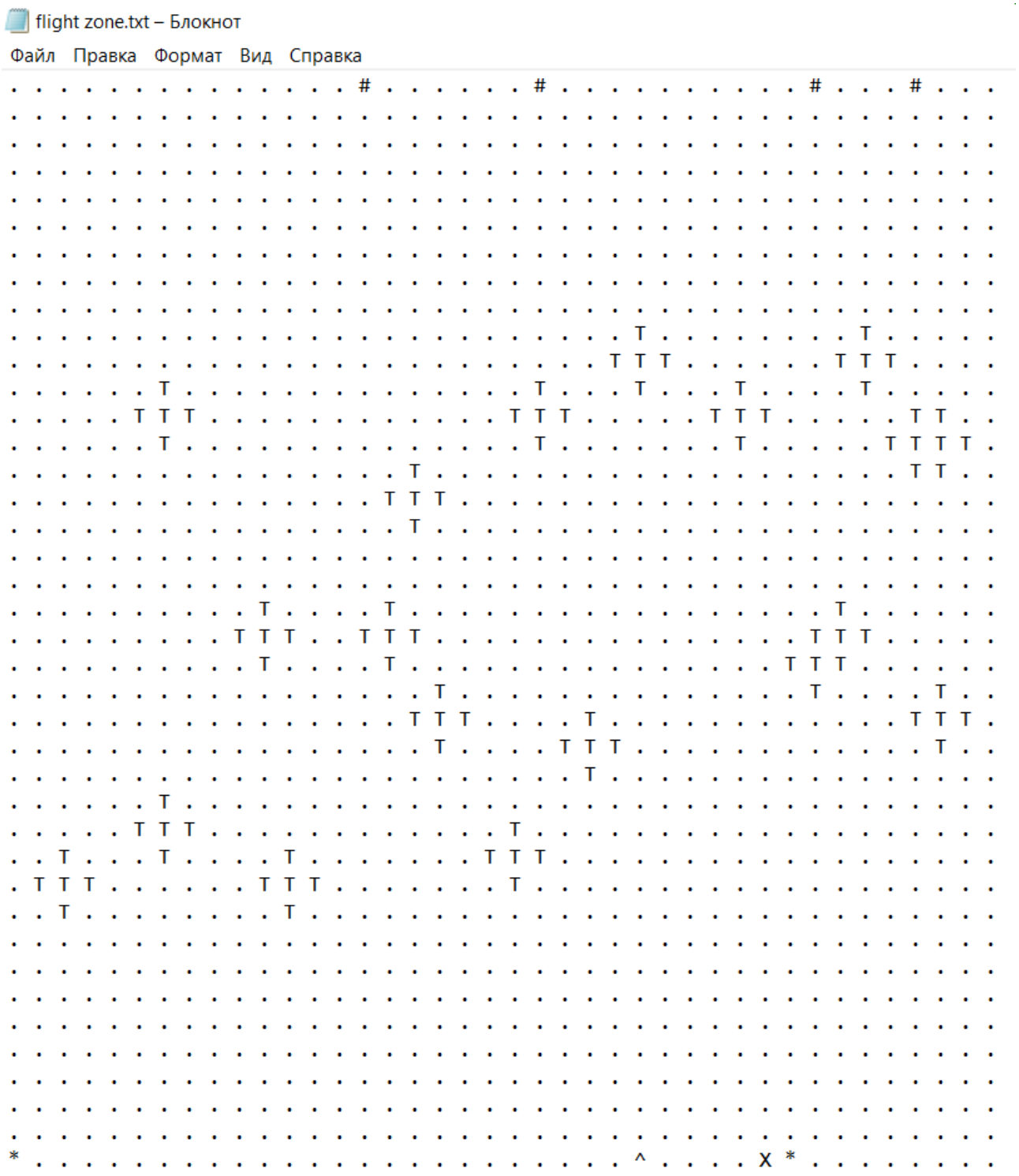

Рисунок 3.2.17 - Відповідна йому зона симуляції

3.3 Додаткова інформація

Важливо зазначити що цей проект є багато файловим, що в свою чергу має свої плюси, такі як:

- Більш логічно розділити різні частини симуляції, що в свою чергу спрощує організацію всього проекту. Полегшення розуміння коду також є доволі важливим додатком до цього. Також кожен файл містить свої власні функції та класи тим самим забезпечуючи відповідну ієрархію та легкість розширення коду проекту.
- Є можливість повторно використовувати певні функції, класи, бібліотеки тощо, адже їх можна створити в одному окремому файлі та після цього використовувати їх в різних частинах проекту.
- Також це допомагає при тестуванні, адже можна тестувати окремі функції, що в свою чергу веде до того що після знаходження помилки можна буде досить швидко знайти її причину.

Також важливо зазначити що всі типи дронів походять від батьківського класу Drone, тобто вони всі мають деякі однакові класи, але також кожен з них має також свої унікальні функції які відповідають їх можливостям.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи було розроблено та реалізовано інформаційну систему симуляції польоту рою дронів. Система покликана дати базове розуміння польоту рою дронів, а також дати приблизне графічне розуміння польоту.

При виконанні було проаналізовано аналогічні продукти та близькі продукти до теми, такі як симулятори навчання управлінням дронів.

Було проаналізовано мови програмування та середовища розробки, та після цього було вирішено скористатися мовою C++ оскільки вона є доволі продуктивною, надає доволі широкі інформаційні можливості а також має доволі великою кількістю стандартних бібліотек. Щодо середовища програмної розробки було обрано CLion, оскільки ознайомленість з його аналогом для мови програмування Java є найбільшою.

При виконанні було проаналізовано та створено графічне відображення зони польоту що в свою чергу дозволяє наглядно показувати переміщення, що в свою чергу спрощує розуміння цього процесу.

Також було створено меню вибору різних перемінних для того аби налаштувати симуляцію під потрібні умови.

Також було вирішено що проект буде багато файловим, адже це допомагає більш логічно розділяти код на сегменти, що в свою чергу полегшить його розуміння та організацію. Ще один плюс в тому що при зміни або редагуванні файли майже не впливають один на одного.

Надалі планується створити ще більше перемінних для того аби можна було налаштовувати симуляцію більш точно. Також планується створити більш краще меню вибору. Також планується знайти та адаптувати більш відповідний алгоритм пошуку шляху, для того аби дрони могли оминати перешкоди та відповідно в кожній симуляції потрапляти до кінцевої точки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. DJI Flight Simulator - DJI. *DJI Official*. URL: https://www.dji.com/global/simulator (дата звернення: 18.05.2023).

2. DJI Virtual Flight - Download-Center - DJI. *DJI Official*. URL: https://www.dji.com/global/downloads/djiapp/dji-virtual-flight (дата звернення: 18.05.2023).

3. Zephyr Drone Simulator. *Zephyr Drone Simulator*. URL: https://zephyr-sim.com/ (дата звернення: 18.05.2023).

4. DOU: Рейтинг мов програмування 2023. URL: https://dou.ua/lenta/articles/language-rating-2023/ (дата звернення 20.05.2023)

5. CLion: умная кросс-платформенная IDE JetBrains для C и C++. *JetBrains*. URL: https://www.jetbrains.com/ru-ru/clion/ (дата звернення: 18.05.2023).

6. Visual Studio: IDE and Code Editor for Software Developers and Teams. *Visual Studio*. URL: https://visualstudio.microsoft.com (дата звернення: 18.05.2023).

7. Notepad++. *Notepad++*. URL: https://notepad-plus-plus.org/ (дата звернення: 20.05.2023).

8. A* Shortest Path - Neo4j Graph Data Science. *Neo4j Graph Data Platform*. URL: https://neo4j.com/docs/graph-data-science/current/algorithms/astar/ (дата звернення: 21.05.2023).

9. Depth First Search or DFS for a Graph - GeeksforGeeks. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/depth-first-search-or-dfs-for-a-graph/ (дата звернення: 21.05.2023).

10. Breadth First Search or BFS for a Graph - GeeksforGeeks. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/ (дата звернення: 21.05.2023).

11. Sneh. Two Dimensional Array in C++. *DigitalOcean | The Cloud for Builders*. URL: https://www.digitalocean.com/community/tutorials/two-

dimensional-array-in-c-plus-plus (дата звернення: 01.06.2023).

12. Учасники проектів Вікімедіа. Алгоритм пошуку A* – Вікіпедія. *Вікіпедія*. URL: https://uk.wikipedia.org/wiki/Алгоритм_пошуку_A* (дата звернення: 22.05.2023).

13. Contributors to Wikimedia projects. Depth-first search - Wikipedia. *Wikipedia, the free encyclopedia*. URL: https://en.wikipedia.org/wiki/Depth-first_search (дата звернення: 22.05.2023).

14. Contributors to Wikimedia projects. Breadth-first search - Wikipedia. *Wikipedia, the free encyclopedia*. URL: https://en.wikipedia.org/wiki/Breadth-first_search (дата звернення: 22.05.2023).

15. Microsoft C/C++ Documentation. *Microsoft Learn: Build skills that open doors in your career*. URL: https://learn.microsoft.com/en-us/cpp/?view=msvc-160 (дата звернення: 30.04.2023).

16. Как задать размер окна консольного приложения c++. *Stack Overflow на русском*. URL: https://ru.stackoverflow.com/questions/41324/Как-задать-размер-окна-консольного-приложения-с (дата звернення: 10.05.2023).

17. C++ | Файловые потоки. Открытие и закрытие. *METANIT.COM - Сайт о программировании*. URL: https://metanit.com/cpp/tutorial/8.2.php (дата звернення: 25.05.2023).

ДОДАТОК А
ПОЧАТКОВИЙ ВИГЛЯД КОДУ ПОЛЬОТУ

```
void Drone::Move(char zone[40][40]) {


   if (f_y < 41 && battery_discharge != 0) {


      if (battery == 0) {
         cout << D_num << " type " << D_type << " charge reach 0 and stuck on "
<< currentX << " " << currentY << endl;
      }

      else {
            battery -= battery_discharge;

            if (currentX == destinationX) {
                currentY += velocity;
            }
            else if (currentX > destinationX) {

                if (currentX - destinationX > 1) {
                    currentX -= velocity;
                    currentY += velocity;
                } else if (currentX - destinationX == 1) {
                    if (velocity != 1) {
                        currentX -= velocity / 2;
                        currentY += velocity / 2;
                    } else if (velocity == 1) {
                        currentX -= velocity;
                        currentY += velocity;
                    }

                } else if (currentX - destinationX < 1) {
                    currentX -= velocity;
                    currentY += velocity;

                }

            } else if (currentX < destinationX) {

                if (destinationX - currentX < 1) {
                    currentX += velocity;
                    currentY += velocity;
                } else if (destinationX - currentX == 1) {

                    if (velocity != 1) {
                        currentX += velocity / 2;
                        currentY += velocity / 2;
                    } else if (velocity == 1) {
                        currentX += velocity;
                        currentY += velocity;
                    }

                } else if (destinationX - currentX > 1) {
                    currentX += velocity;
                    currentY += velocity;
                }
```

```
            }

            if (currentY > 40) {
                currentY = 40;
            }

         if (zone[currentX][currentY] == '1') {
            battery += 25;
            cout << "Battery charged + 25" << endl;
         }

         else if (zone[currentX][currentY] == '2') {
            battery += 15;
            cout << "Battery charged + 15" << endl;
         }

         else if (zone[currentX][currentY] == '3') {
            battery += 5;
            cout << "Battery charged + 5" << endl;
         }

         cout << D_type << " with id " << D_num << " located in " << currentX <<
" " << currentY << endl;

         cout << "Curent destination " << destinationX << " " << f_y << endl;

         if (currentX == destinationX && currentY == f_y) {
            cout << D_type << " drone with id " << D_num << " reach destination"
<< endl;
            battery += battery_discharge;
         }

      }

   }
}
```

ДОДАТОК Б
ПСЕВДО АНІМАЦІЯ

```
void intro() {
   char key = TOP;
   string str = "><DRONE DELIVERY><";
   cout << setw(centerString(str)) << "";
   thread opening(menuSound);
   smoothPrint(str, 50);
   opening.detach();
   setcur(0, 0);
   str = "     ><DRONE DELIVERY><     ";
   int c = centerString(str);
   while (key != ENTER) {
      if (!_kbhit()) {
         cout << setw(c) << "" << "     ><DRONE SIMULATION><     " << endl;
         cout << setw(c) << "" << "                              " << endl;
         cout << setw(c) << "" << "       ~~.~~      ~~.~~       " << endl;
         cout << setw(c) << "" << "    ~~.~~ \\\|__::__|/ ~~.~~    " << endl;
         cout << setw(c) << "" << "       \\___\\__||__/___/       " << endl;
         cout << setw(c) << "" << "        //          \\\\        " << endl;
         cout << setw(c) << "" << "                              " << endl;
         cout << setw(c) << "" << "   [PRESS ENTER TO START]     " << endl;
         Sleep(100);
         setcur(0, 0);
         cout << setw(c) << "" << "     <>DRONE SIMULATION<>     " << endl;
         cout << setw(c) << "" << "                              " << endl;
         cout << setw(c) << "" << "        ~.~      ~.~         " << endl;
         cout << setw(c) << "" << "     ~.~  \\\|__::__|/  ~.~     " << endl;
         cout << setw(c) << "" << "       \\___\\__||__/___/       " << endl;
         cout << setw(c) << "" << "        //          \\\\        " << endl;
         cout << setw(c) << "" << "                              " << endl;
         cout << setw(c) << "" << "   [PRESS ENTER TO START]     " << endl;
         Sleep(100);
         setcur(0, 0);
      }
      else {
         key = _getch();
      }
   }
   system("cls");
   menuSound();
}
```

ДОДАТОК В
BUILDING

```
class Building {
private:
   char sign = 'B';
   int buildingX;
   int buildingY;
    int lenght;
    int widht;

public:
   void setBuilding(char area[][AREA_SIZE]);
   bool isBuilding(int x, int y, char area[][AREA_SIZE]);
};
```

```
void Building::setBuilding(char (*area)[AREA_SIZE]) {
    buildingX = AREA_SIZE / 4 + rand() % (AREA_SIZE / 2);
    buildingY = 1 + rand() % (AREA_SIZE - 2);
    lenght = 1 + rand() % 5;
    widht = 1 + rand() % 5;
    for (int i = buildingX;i<=buildingX + lenght;i++){
        for(int j = buildingY;j>=buildingY - widht; j--){
            area[i][j] = sign;
        }
    }
}

bool Building::isBuilding(int x, int y, char (*area)[AREA_SIZE]) {
        for (int i = buildingX;i<=buildingX + lenght;i++){
            for(int j = buildingY;j>=buildingY - widht; j--){
                if(area [i][j]!=sign){
                    area[i][j] = sign;
                    return true;
                }
            }
        }
        return false;
}
```

ДОДАТОК Г
TREE

```
class Trees {
private:
    char sign = 'T';
    int treeX;
    int treeY;

public:
    void setTrees(char area[][AREA_SIZE]);
    bool isTree(int x, int y, char area[][AREA_SIZE]);
};
```

```
void Trees::setTrees(char (*area)[AREA_SIZE]) {
    treeX = AREA_SIZE / 4 + rand() % (AREA_SIZE / 2);
    treeY = 1 + rand() % (AREA_SIZE - 2);

    area[treeX][treeY] = sign;
    area[treeX][treeY + 1] = sign;
    area[treeX][treeY - 1] = sign;
    area[treeX + 1][treeY] = sign;
    area[treeX - 1][treeY] = sign;
}

bool Trees::isTree(int x, int y, char area[][AREA_SIZE]) {
    if (area[treeX][treeY] != sign) {
        area[treeX][treeY] = sign;
        return true;
    }
    if (area[treeX][treeY + 1] != sign) {
        area[treeX][treeY + 1] = sign;
        return true;
    }
    if (area[treeX][treeY - 1] != sign) {
        area[treeX][treeY - 1] = sign;
        return true;
    }
    if (area[treeX + 1][treeY] != sign) {
        area[treeX + 1][treeY] = sign;
        return true;
    }
    if (area[treeX - 1][treeY] != sign) {
        area[treeX - 1][treeY] = sign;
        return true;
    }
    return false;
}
```

ДОДАТОК Д
НОВИЙ КОД ДЛЯ РУХУ ДРОНІВ

```
bool Drone::move(char area[][AREA_SIZE]) {
    if (batteryCharge > 0) {
        if (currentX != destinationX) {
            if (currentY == destinationY) {
                area[currentX][currentY] = '.';
                area[currentX -= velocity][currentY] = sign;
                batteryCharge -= batteryDischarge;
            } else if (currentY > destinationY) {
                if (currentX - destinationX > 1) {
                    area[currentX][currentY] = '.';
                    area[currentX -= velocity][currentY -= velocity] = sign;
                    batteryCharge -= batteryDischarge;
                } else if (currentY - destinationY == 1) {
                    if (velocity > 1) {
                        area[currentX][currentY] = '.';
                        area[currentX -= (velocity / 2)][currentY -= (velocity /
2)] = sign;
                        batteryCharge -= batteryDischarge;
                    } else if (velocity == 1) {
                        area[currentX][currentY] = '.';
                        area[currentX -= velocity][currentY -= velocity] = sign;
                        batteryCharge -= batteryDischarge;
                    }
                } else if (currentY - destinationY < 1) {
                    area[currentX][currentY] = '.';
                    area[currentX -= velocity][currentY -= velocity] = sign;
                    batteryCharge -= batteryDischarge;
                }
            } else if (currentY < destinationY) {
                if (destinationY - currentY < 1) {
                    area[currentX][currentY] = '.';
                    area[currentX -= velocity][currentY += velocity] = sign;
                    batteryCharge -= batteryDischarge;
                } else if (destinationY - currentY == 1) {
                    if (velocity != 1) {
                        area[currentX][currentY] = '.';
                        area[currentX -= velocity / 2][currentY += velocity / 2]
= sign;
                        batteryCharge -= batteryDischarge;
                    } else if (velocity == 1) {
                        area[currentX][currentY] = '.';
                        area[currentX -= velocity][currentY += velocity] = sign;
                        batteryCharge -= batteryDischarge;
                    }
                } else if (destinationY - currentY > 1) {
                    area[currentX][currentY] = '.';
                    area[currentX -= velocity][currentY += velocity] = sign;
                    batteryCharge -= batteryDischarge;
                }
            }
            if (currentX < 0) {
                currentX = 0;
                area[currentX][currentY] = sign;
            }
        }
    }
}
```