Міністерство освіти і науки України
Національний аерокосмічний університет ім. М. Є. Жуковського
«Харківський авіаційний інститут»

Факультет систем управління літальних апаратів

Кафедра систем управління літальних апаратів

Пояснювальна записка

до дипломної роботи
магістра
(освітньо-кваліфікаційний рівень)

на тему Розробка і дослідження інформаційної системи визначення маршруту польоту безпілотного літального апарату для проведення аерофотозйомки

ХАІ.301.361.24О.272.00183013 ПЗ

Виконала: студентка 2 курсу, групи 361

Галузь знань 27 «Транспорт»

Спеціальність

272 “Авіаційний транспорт”

Освітня програма

“Інтелектуальні транспортні системи”

Цибулько Марина Вікторівна
(прізвище та ініціали студента)

Керівник Свищ В. М.
(прізвище та ініціали)

Рецензент Жученко О. С.
(прізвище та ініціали)

Харків – 2024

Міністерство освіти і науки України
Національний аерокосмічний університет ім. М. Є. Жуковського
«Харківський авіаційний інститут»

Факультет __систем управління літальних апаратів_____
(повне найменування)
Кафедра ___систем управління літальних апаратів_____
(повне найменування)
Рівень вищої освіти _____другий (магістерський)_____
Галузь знань _____27 «Транспорт»_____
(код та найменування)
Спеціальність _____272 “Авіаційний транспорт”_____
(код та найменування)
Освітня програма __“Інтелектуальні транспортні системи”_____
(найменування)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____Костянтин ДЕРГАЧОВ
(підпис) (ім’я та прізвище)
«___»_____ 2024 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

_____Цибулько Марин Вікторівна_____
(прізвище, ім’я та по батькові)
1. Тема кваліфікаційної роботи «Розробка і дослідження інформаційної системи визначення маршруту польоту безпілотного літального апарату для проведення аерофотозйомки» _____
керівник кваліфікаційної роботи
д-р. техн. наук, професор Свищ Володимир Митрофанович_____
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом Університету № 1968-уч від «06» листопада 2023 року.
2. Термін подання здобувачем кваліфікаційної роботи « 08 » січня 2024 р.
3. Вихідні дані до роботи Математична модель руху БПЛА, вимоги до функціонування системи та проведення аерофотозйомки_____
4. Зміст пояснювальної записки (перелік завдань, які потрібно розв’язати) __ Вступ, аналіз проблеми, опис об’єкту, математичний опис об’єкту, синтез середовища для моделювання, конструкторська частина, розробка дослідницького процесу, експериментальна частина, економічна частина, заключення, список використаних джерел_____
5. Перелік графічного матеріалу __10 плакатів формату А1 у вигляді слайдів

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Стан проблеми	проф., д.т.н. Свищ В.М.	12.09.23	10.01.24
Аналіз даної задачі	проф., д.т.н. Свищ В.М.	12.09.23	10.01.24
Конструктор. частина	проф., д.т.н. Свищ В.М.	12.09.23	10.01.24
Дослідницька частина	ст. викл. Бичкова І.В.	12.09.23	10.01.24
Експ.-практ. частина	ст. викл. Бичкова І.В.	12.09.23	10.01.24
Економічне обгрунтування	ст. викл. Бичкова І.В.	12.09.23	10.01.24

Нормоконтроль _____ __Свищ В.М.__ «_16_» ____01_____2024 р.
(підпис) (ім'я та прізвище)

7. Дата видачі завдання «_12_» _вересня_2023_ р.

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Формулювання теми проекту.	08.09.2023	
2	Розробка технічного завдання	12.09.2023	
3	Огляд стану проблеми і патентний пошук. Математичний опис системи управління.	06.10.2023	
4	Аналіз предметної області	20.10.2023	
5	Конструкторська частина роботи. Дослідницька частина роботи. Економічне обґрунтування розробки	22.12.2023	
6	Експериментально-практична частина	26.12.2023	
7	Оформлення розрахунково-пояснювальної записки і графічного матеріалу. Попередній захист роботи	08.01.2024	
8	Рецензування проекту. Захист проекту в ЕК		

Здобувач _____ Марина_ЦИБУЛЬКО
(підпис) (ім'я та прізвище)

Керівник кваліфікаційної роботи

_____ Володимир СВИЩ
(підпис) (ім'я та прізвище)

Міністерство освіти і науки України
Національний аерокосмічний університет ім. М. Є. Жуковського
«Харківський авіаційний інститут»

Кафедра систем управління літальних апаратів (№301)

«ЗАТВЕРДЖУЮ»
Завідуючий кафедрою 301
к.т.н., с.н.с., доцент
_____Костянтин ДЕРГАЧОВ
«___» _____2024 р.

ТЕХНІЧНЕ ЗАВДАННЯ
на дипломне проектування
Цибулько Марині Вікторівні

1 Тема роботи: Розробка і дослідження інформаційної системи визначення маршруту польоту безпілотного літального апарату для проведення аерофотозйомки
затверджена наказом по університету від «06» листопада 2023 р. № 1968-уч .
2 Строк здачі студентом закінченої роботи « 08 » січня 2024 р.
3 Область застосування розробки: робота з нестійкими космічними об'єктами
4 Початкові дані для розроблювальної системи
4.1 Призначення і мета створення системи: створення ефективної системи визначення польоту для проведення аерофотозйомки
4.2 Загальні відомості завдання на дипломний проект, програмні аналоги для відображення керованих рухомих графічних об'єктів
5 Технічні вимоги до каналів системи управління
5.1 Питання, що підлягають розробці: формування математичної моделі, вибір виконавчого та вимірювальних пристроїв, синтез напівнатурного моделювання.
5.2 Режим роботи системи (безперервний, циклічний, одноразової дії): автоматичний
6 Умови експлуатації системи:
6.1 Кліматичні вимоги до експлуатації (температура середовища, у якій буде працювати система управління, її вологість, вміст хімічно активних компонентів і т.ін.): температура середовища ±20 градусів за Цельсієм.
6.2 Механічні вимоги (вібрація, тряска, можливі перекоси, удари, нахили і т.ін.): можливі вібрація, удари, нахили.
6.3 Наявність перешкод (електричні наведення радіоперешкоди, магнітні впливи): можна знехтувати
7 Додаткові функції, реалізовані системою (сигналізація про несправності, реєстрація необхідної інформації, самоконтроль самої системи і т.ін.): Дана система не має додаткових функції.
8 Обсяг виконуваних розроблювачем робіт
8.1 Етапи проведення роботи: I етап: Оцінка стану проблеми.
II етап: Вербальний і математичний опис об'єкту

III етап: Синтез пристрою

8.2 Обсяг розробки по кожному етапу:

1) стан проблеми і постановка задачі проектування (10 ст)

2) опис задачі та математичний опис (30 ст)

3) конструкторська частина (9 ст)

4) розробка дослідницького процесу (18 ст)

5) економічна частина (11 ст)

9. Параметри обладнання системи

9.1. Габарити розміри визначаються у процесі проектування

9.2. Маса визначається в процесі проектування

9.3. Вимоги до конструктивного виконання і розміщенню визначаються в процесі проектування

9.4. Інші вимоги не передбачаються

10. Вимоги безпеки визначаються в процесі проектування

11. Дослідницька частина

11.1. Розробка дослідницького процесу розробка технології дослідження контролера з периферійним обладнанням

11.2. Умови і вимоги технологія перевірки апаратних засобів

11.3. Очікуваний результат створення повністю працездатної та повнофункціональної моделі

12. Економічна частина

12.1. Розробити (розрахувати, отримати): трудомісткість виконання роботи, розрахунок витрат на виконання роботи, договірна ціна роботи.

12.2. Умови і вимоги одиничне виробництво, розрахунок проводиться за статтями калькуляції

12.3. Очікуваний результат собівартість виробу грн. з урахуванням усіх витрат

13. Перелік графічних матеріалів та їх формат 10 плакати формату А4:

1) опис об’єкту автоматичного управління, математична модель об’єкту, результат дослідження об’єкту.

2) середовище розробки.

3) дослідницька частина, економічна частина.

4) висновок.

14. Мова підготовки пояснювальної записки (захисту) українська

Керівник роботи	Прийняв до виконання
проф., д.т.н. Свищ В.М.	Цибулько М. В.
(П.І.Б.)	(П.І.Б. студента)
«12» вересня 2023 р.	«12» вересня 2023 р.

Погоджено з питань:

конструкції
_____проф., д.т.н. Свищ В.М._____
(П.І.Б.)
_____
«12» вересня 2023 р.

дослідницької частини
_____ст.викл. Бичкова І. В._____
(П.І.Б.)
_____
«12» вересня 2023 р.

економіки
___ ст.викл. Бичкова І. В._____
(П.І.Б.)
_____
«12» вересня 2023 р.

РЕФЕРАТ

Пояснювальна записка містить: 101 стор., 31 рис., 7 табл., 1 дод., 21 джерел.

Мета роботи: розробити інформаційну модель для визначення маршруту безпілотного літального апарату в місці проведення аерофотозйомки.

Об'єкт дослідження: безпілотний літальний апарат, який рухається до місця проведення аерофотозйомки.

Предмет дослідження: рух безпілотного літального апарату в заданому просторі.

Основний зміст роботи: у дипломній роботі було розроблено систему інформаційного моделювання та дослідження траєкторії руху безпілотних літальних апаратів різної конфігурації у змодельованому середовищі.

Отримані результати: інформаційна модель виявлення руху безпілотного літального апарату в місці проведення аерофотозйомки.

Ключові слова: БПЛА, РОЗРОБКА ПРОГРАМ НА C#, РОЗРОБКА НА UNITY, СИМУЛЯЦІЯ РОБОТИ РЛС, SFML

СПИСОК СКОРОЧЕНЬ І УМОВНИХ ПОЗНАЧЕНЬ

SFML – Simple and Fast Multimedia Library

VS2017 – Visual Studio 2017

MIT – Massachusetts Institute of Technology

БПЛА – безпілотний літальний апарат

MPEI – макет розташування елементів інтерфейсу

ОС – операційна система

GUI (Graphical User Interface) — графічний інтерфейс користувача

ЗМІСТ

ВСТУП

Бурхливий розвиток безпілотної авіації в останні роки змінив організацію моніторингу статичних і рухомих об'єктів. Це пояснюється наступними причинами: безпілотні літальні апарати (БПЛА) значно дешевші за пілотовані літаки-розвідники; БПЛА легко контролювати та контролювати; БПЛА можуть працювати вночі, в умовах поганої видимості, в ситуаціях, коли життя пілота знаходиться під загрозою. Завдання виявлення рухомих об'єктів на землі не нова. Однак цей тип рельєфу має ознаки одноступінчастої зони через наявність значних площ лісу. У таких зонах важко виявити об'єкти спостереження. Особливо складним є процес вибору оптимальної траєкторії польоту. При цьому йдеться не лише про виконання цивільних завдань, а й про вирішення військових завдань.

Останнім часом характер і методи ведення збройних конфліктів змінилися. На перший план виходить бойове застосування високомобільних малих груп, здатних виконувати терористичні завдання. Сучасний тероризм – складне соціально-політичне явище як форма політичного конфлікту, форма поведінки окремих осіб та організованих груп. Вони, порушуючи громадський порядок, дестабілізують процес суспільного життя, сприяють створенню та розвитку конфліктних ситуацій. Протидія терористичним атакам таких угруповань потребує достовірної та своєчасної інформації. Це можна швидко зробити, проводячи активну розвідку (спостереження). Планування маршруту повітряного пошуку з БПЛА в лісостеповій зоні потребує виявлення та врахування особливих закономірностей динамічних дій об’єктів. Сам план розвідки повинен передбачати хід дій противника.

Прогнозування можливих динамічних дій об'єктів буде результатом, якщо відома інформація про їх початкове положення (час, координата, положення). У цьому випадку результати аеропошуку за допомогою БПЛА нададуть інформацію про напрямок руху, можливі цілі та тимчасові рухомі об’єкти, що залишають ціль. Ця інформація дозволить своєчасно підготувати заходи щодо протидії можливим терористичним актам.

Проблема спостереження за статичними та динамічними об’єктами полягає в тому, що існує конфлікт між розміром зони виявлення та розміром об’єктів. Вирішення цього протиріччя можливе за рахунок автономної роботи БПЛА, зміни його траєкторії польоту (польотного завдання) за результатами автоматичного розпізнавання об’єктів. Сучасний БПЛА містить автопілот і потужний бортовий комплекс, який може виконувати ці завдання за умови оснащення відповідними інформаційними технологіями. Таким чином, відповідним завданням є розробка інформаційних технологій автоматичного виявлення та ідентифікації нерухомих об’єктів безпілотними літальними апаратами.

1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Значення та актуальність БПЛА

Створення безпілотних літальних апаратів зараз перетворюється з вузькоспеціалізованої сфери розробки і виробництва у велику індустрію, яка має великий потенціал для багатьох сфер людської діяльності. Слід зазначити, що такі безпілотники вже можуть приносити переваги, переміщуючись у повітряному просторі та дистанційно взаємодіючи з системою управління, але з різними специфічними завданнями все дещо складніше.

Багато сучасних БПЛА оснащені не лише апаратними компонентами, необхідними для польоту, живлення та зв’язку, а й вдосконаленим програмним компонентом, який не лише обслуговує систему безпілотників на базовому рівні, але й деякі також спеціалізуються на інтелектуальних завданнях – такими є БПЛА. . Спеціальні програмні засоби для тієї чи іншої діяльності в їх структурі.

Звичайно, перше, що спадає на думку, це універсальне рішення - установка на такі машини системи штучного інтелекту, яку потім можна буде навчити виконувати різні завдання, крім того - самокоригувати її алгоритм. Така система може значно заощадити гроші та час, а також зробити дрони універсальними за активністю. Але справа в тому, що розробка ШІ, хоч і має великі успіхи, але, по-перше, поки що не дозволяє всебічно тренувати в необхідному обсязі, який буде потрібно для швидкого «перенавчання» дронів, а по-друге, та сама система . З універсальним ШІ бортова обробка даних вимагатиме не лише відносно ефективного (і дорогого) апаратного компонента, а й адекватного джерела живлення, тобто ефективного елемента живлення, який у сучасних реаліях і гріє (проблеми з охолодженням системи), і дронить. (через недостатню легкість такого елемента). Звичайно, зараз розробляються прототипи з іншими силовими

елементами (наприклад, водневе паливо використовується замість поширених літій-іонних або літій-полімерних акумуляторів), але на даний момент при масовому виробництві БПЛА критичний баланс важливий[1]. Необхідно підтримувати між масою, маневреністю та дизайном, а також як результат автономної роботи.

Отже, поки універсальний штучний інтелект і збалансоване джерело живлення все ще знаходяться в розробці, давайте повернемося до ідеї спеціалізованих програмних засобів. Таке програмне забезпечення розробляється для конкретних предметних галузей, а також для окремих завдань і добре класифіковано. Не дивно, що таке програмне забезпечення також розробляється для БПЛА в залежності від майбутньої діяльності таких машин. Важливо відзначити, що остаточна обробка конкретних даних не обов'язково повинна виконуватися самим дроном, витягнуті дані будуть передані в наступний блок обробки, для якого більше не буде автономної роботи або відносно обмеженої обчислювальної потужності. . Критичний елемент, який лише завадить таким діям сил безпілотників. Але враховуючи такий процес збору, передачі та обробки даних, важливо, щоб усі ланки обробки працювали рівномірно та безпомилково – вихідні дані, отримані дроном, перетворюються в остаточну форму, і це все правильно. Комплексні рішення спеціальних програмних засобів (СПЗ), що працюють в БПЛА та інших компонентах (за наявності).[1]

Також можна перерахувати наступні основні напрямки розробки SDR для дронів:

1. Управління зв'язком між БПЛА принаймні дозволяє дронам уникати зіткнень між ними шляхом завчасного попередження про їхню присутність, або на повній дистанції доступ до групи дронів дозволить вам створити ціль мережу з БПЛА, які можуть формувати «рій». Це програмне забезпечення розроблено компаніями Airware і Yuneec за підтримки Intel. AT&T також бере участь в інтеграції мереж 4G для обміну інформацією між дронами.

2. Виявлення та запобігання зіткненням може не включати двосторонній обмін інформацією для коригування операцій. Крім того, різні країни матимуть різні правила повітряного простору, а деякі вимагатимуть інтеграції БПЛА в системи управління повітряним рухом. Airware і PixiePath розробляють відповідні SDR. Таке програмне забезпечення забезпечує не тільки ручні системи управління та аналізу, але й алгоритми ШІ для автономної роботи дронів. У цьому документі розроблено модель, яка конкретно стосується цього питання [18].

3. Обробка витягнутих даних. Як уже було сказано, це тип даних, який не є критичним для систем дронів, оскільки для такої базової навігації SDR розробляється з пункту 2. Наприклад, Pix4D розробив SDR, який допомагає безпілотникам виконувати безперервні топографічні дослідження. У різних випадках він перетворює масив зображень під час польоту в двовимірні зображення та тривимірні моделі місцевості

Можна точно сказати, що дрони мають великий потенціал, адже вже зараз помітна їхня надзвичайна продуктивність, а питання такого розвитку актуальне як ніколи, особливо зараз – під час пандемії, коли важливі методики дистанційного зв’язку.

1.2. Види дронів.

Для простоти розуміння ми не будемо розповідати про всі нюанси створення і призначення систем БПЛА, які прийняті і використовуються в усьому світі [20]. Ці дані легко знайти в Інтернеті. Ми обмежуємося вступною лекцією, яка дозволяє бажаючим знайти всю необхідну інформацію та відповіді на запитання в Інтернеті.

Для початку розглянемо тип безпілотників, які з кожним днем стають все більш поширеними – багатороторні системи. Залежно від кількості пропелерів їх ще називають мультикоптер, квадрокоптер, гексакоптер, октакоптер тощо. Однією з відмінних рис багатомоторної системи є схожість принципу польоту з вертолітним. Перевагами цієї платформи є відсутність

потреби в домашній базі для зльоту та посадки, можливість плавати на одному місці та простота керування.

Недоліки, які обмежують використання коптерів: малий радіус дії, неможливість використання при сильному вітрі, висока чутливість до замерзання, необхідність установки більших акумуляторів, ніж системи літака. Багатороторні системи зазвичай працюють на відстані 10 км (вертольоти основної маси до 4 км) у спокійному нерухомому повітрі. Робоча висота не змінюється більше ніж від 250 до 800 метрів в залежності від встановленого обладнання спостереження. Вони дуже ефективні в міській забудові та дозволяють зазирнути за топографію території чи будівлі. Комфортні коптери в корегуванні вогню артилерії - в режимі висіння. Часто їх використовують для пошуку ДРГ біля окопів у темний час доби, якщо БПЛА оснащений тепловізором. Діапазон робочих швидкостей, як правило, до 10 м/с. Маленькі коптери в режимі ручного керування можуть розганятися до 20 метрів за секунду.

Другим за популярністю [2], але не найефективнішим типом дронів є тип літака. Перевагами цих систем є велика дальність, більша енергоефективність порівняно з коптерами та менша залежність від погоди. Відстань, яку проходить авіаційний безпілотник найпростішого класу – «бойові поля» – у кілька разів перевищує робочу дальність коптерних систем. Недоліки БПЛА авіаційного базування: необхідність у злітно-посадкових майданчиках, більший час розгортання та підготовки, більш складне управління та складніша підготовка екіпажу. Вони використовуються для аерофотозйомки вдень і вночі, а за наявності в екіпажу необхідних навичок – для коригування вогню артилерії.

БПЛА призначені для виконання завдань РЕР, EW та зв’язку. Діапазон робочих швидкостей від 15 до 30 м/с. Робоча висота - залежно від комплектації та розміру машини, але завжди більше 300 метрів. Зазвичай цей діапазон висот становить 300-2000 метрів. Існує кілька аеродинамічних конструкцій авіаційних дронів. Основні аеродинамічні конструкції – класичні та «літаючі крила».

Центрування літака - це розташування центру ваги літака відносно хорди крила. Весь вантаж на БПЛА повинен бути розміщений таким чином, щоб центрування не виходило за межі центрів, дозволених для конкретного повітряного судна. Діапазон осей для літаків класичного компонування становить від 25 до 35% SAH. SAH – середня аеродинамічна хорда крила. Для крила прямокутної форми це хорда, для крил складної форми вона визначається розрахунком.

Виробники БПЛА зазвичай вказують розташування центру ваги для зручності екіпажу, часто в міліметрах від передньої кромки крила, а найдосконаліші позначають центральну лінію на крилі. Дрон, що навшпиньки проходить над центральними мітками, повинен бути в нейтральному балансі. Перед кожним зльотом необхідно перевіряти центрування. Наприклад, неточність установки батареї може істотно змінити центр.

Це збільшує втрати енергії для стабілізації дрона під час польоту або навіть аварії відразу після зльоту [3].

Серед типів аеродинамічних конструкцій можна виділити кілька основних конструкцій, які є найбільш популярними для БПЛА: класична конструкція з опорним гвинтом, класична конструкція з гвинтом-штовхачем, закрилка з штовхачем, закрилка з гвинтом.

Рисунок 1.1 - Класична схема з тягнучим гвинтом

Рисунок 1.2 - Класична схема зі штовхальним гвинтом

Рисунок 1.3 - Літаюче крило зі штовхальним гвинтом

Рисунок 1.4 - Літаюче крило з тягнучим гвинтом

У порівнянні з класичною конструкцією, літаючі крила більш технологічні, легші в транспортуванні, менш схильні до пошкоджень при необережному приземленні. Але збереження фюзеляжу та хвостового оперення робить літаючі крила менш стабільними з точки зору тангажу та траєкторії - показано мале плече кермових поверхонь щодо центру ваги

літака. Літаюче крило не може стартувати як БПЛА класичної конструкції, тому для надійного старту потрібен більш потужний двигун.

БПЛА класичної конструкції, хоч і вимагають більшої точності при транспортуванні, але демонструють найкращу якість польоту та стабільність у польоті – це важливо для нормальної роботи оптики. Гвинт гвинта-штовхача має додаткові переваги в порівнянні з гвинтом-штовхачем - він не знаходиться в аеродинамічній тіні, як гвинт-штовхач, створюючи додаткові махи в крилі та покращуючи несучі властивості та стійкість.

1.3. Аналіз наявних досліджень і постановка задачі проектування

Говорячи про функцію планування траєкторії польоту БПЛА, зауважте, що переважна більшість публікацій (в основному зарубіжних) використовують такі терміни, як «планування траєкторії» або «побудова траєкторії» [19]. Як правило, ці функції розглядають динаміку руху БПЛА або проблеми мобільності в різних процесах маневрування. Ні, скоріше маючи на увазі проблему побудови траєкторії польоту, яка є самостійною проблемою. Статей, які так чи інакше обговорюють процес складання оптимальної траєкторії протягом усього польоту БПЛА, значно менше.

На сьогоднішній день опубліковано значну кількість робіт, присвячених різним аспектам побудови та цільового застосування БПЛА. Відносно високий рівень розвитку цього наукового напряму відображає низка оригінальних праць узагальнювального та концептуального характеру. Розглянуто питання організації цільової експлуатації БПЛА, у тому числі планування траєкторії польоту, та проблеми, пов'язані з цим

Аналіз опублікованих матеріалів показує, що формалізація задачі планування траєкторії польоту БПЛА є класичною як функція туристичного продавця. Побудова траєкторії польоту зводиться до побудови надпрольотної послідовності точок, положення яких на поверхні землі вважається наперед заданим і виступає критерієм мінімізації довжини

траєкторії. Водночас слід зазначити, що процедура обґрунтування вибору точки за призначенням (завданням) БПЛА в цих роботах не наводиться.

Традиційні формули з'єднують всі задані точки і шукають рішення у вигляді замкнутих траєкторій польоту. Крім того, БПЛА пролітає в кожній точці лише один раз, тобто на шляху немає петель. Таким чином, з математичної точки зору ми називаємо інтерпретацію задачі маршрутизації авіакомпанії класичною закритою задачею продавця туристичних послуг. Згадані функції не враховують можливий вплив погоди на БПЛА під час руху, або швидкість вітру в зоні польоту вважається низькою. Припускаючи відповідь, зменшення довжини маршруту за постійної швидкості вітру дорівнює мінімізації en час польоту по маршруту[4].

Аналізуючи періодичні вітчизняні та зарубіжні літературні джерела, можна зробити висновок, що функції управління та планування використання БПЛА умовно поділяються на три групи (рис. 1.5).

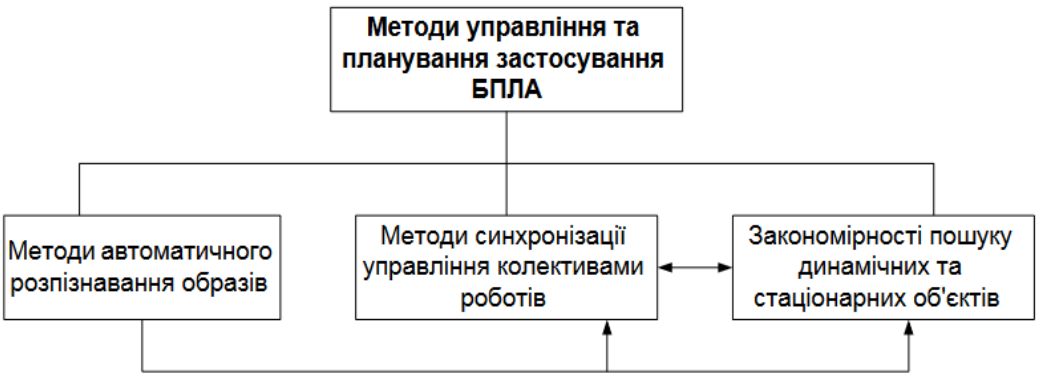


Рисунок 1.5 - Склад методів управління та планування використання БПЛА за результатами аналізу періодичних джерел

Таким чином, сформулюємо задачу проектування за результатами комплексного аналізу, виконаного в першому розділі.

Тому необхідним є розвиток (удосконалення) методів автоматизованого планування траєкторії польоту та інформаційних технологій у системі підтримки прийняття рішень для підвищення ефективності динамічного та статичного виявлення об’єктів БПЛА.

Узагальнена постановка задачі планування траєкторії польоту БПЛА виглядає наступним чином. Оптимально, в певному сенсі, знайти траєкторію польоту БПЛА, яка з'єднує всі або частину точок з відомим місцем розташування, враховуючи інформацію про обмеження, накладені на траєкторію через специфіку цільової ситуації, і обмеження через технічні характеристики БПЛА.

Розвиток ІТ дозволяє включати до складу БпАК систему підтримки прийняття рішень, яка дозволяє оператору приймати рішення щодо маршрутних точок для ефективного виявлення нерухомих об’єктів у русі.

1.4. Висновки за розділом

Наведено аргументи на користь підтримки розробки БПЛА, коротко описано основні труднощі їх створення та напрямки застосування БПЛА.

Аналізуючи переваги та недоліки програмного аналога для візуалізації та керування графічними об’єктами, було зроблено наступні висновки щодо розробки майбутнього проекту, а саме:

- Візуалізація повинна виконуватися у віртуальному 2D просторі, щоб спростити візуалізацію та зменшити навантаження на систему
- Такі програми часто мають перевантажений інтерфейс користувача, тому слід використовувати мінімальний підхід
- Для економії часу при освоєнні драйверів з власним середовищем розробки слід використовувати одну з мов програмування з мультимедійною бібліотекою.
- Візуальний сценарій абсолютно необов'язковий для проекту з відображенням руху дрона
- Розробляти код усіх програмних механізмів візуального відображення з нуля, використовуючи стандартні можливості мови програмування, немає сенсу, для ефективного використання часу потрібно використовувати графічну бібліотеку.

- Драйвер має бути простим і ефективним, тому важливо використовувати графічну бібліотеку, можливо, за допомогою C або C++.

- Таким чином, крім основного завдання, майбутня модель повинна мати зручність у використанні, 2D візуалізацію польоту БПЛА, простий графічний інтерфейс і автоматизацію. Важливо максимально вдосконалити відображення візуальної інформації.

2 АНАЛІЗ ДАННИХ ПРЕДМЕТНОЇ ОБЛАСТІ

2.1. Аналіз переваг і недоліків існуючих програмних аналогів

1) Unreal engine 4

Unreal engine 4— це мультимедійний механізм, який підтримується компанією-розробником Epic Games. Цей движок має чи не найкращий набір опцій, що відповідають за відображення графічних об'єктів (особливо тривимірний віртуальний простір), який підключено до чітко визначеної системи освітлення. Останнім часом багато розробників використовують його для розробки комерційних проектів, спрямованих на парки розваг і різного роду симулятори. Unreal Engine 4 має вбудований візуальний редактор (рис. 1.1), який дозволяє користувачеві взаємодіяти з движком під час розробки на високому рівні, тобто налаштовувати більшість параметрів без ручного написання програмного коду [3]. Простий графічний інтерфейс редактора.

Рисунок 2.1 - Загальний вигляд редактора в Unreal Engine 4

Для логіки додатків движок передбачає два варіанти їх створення: стандартне написання логіки на мові програмування C++ і власну

альтернативу – Blueprint, що є візуальною скриптовою системою (рис. 2.2). Скрипти - це коди, які не потрібно змінювати або видаляти з системи. Програма - це код, який точно представляє цю систему, і при його редагуванні це те, що вимагається від системи. По суті, другий варіант дозволяє використовувати практично весь потенціал програмування через візуальні блоки, тим самим виводячи процес створення логіки програми на ще більш високий рівень, а також створення того чи іншого програмного забезпечення. Підвищити продуктивність застосування. Недоліком даного варіанту є певна неуніверсальність, оскільки креслення як таке використовується тільки в цьому двигуні. Окрім створення тривимірних об’єктів, Unreal Engine 4 також можна використовувати для 2D.

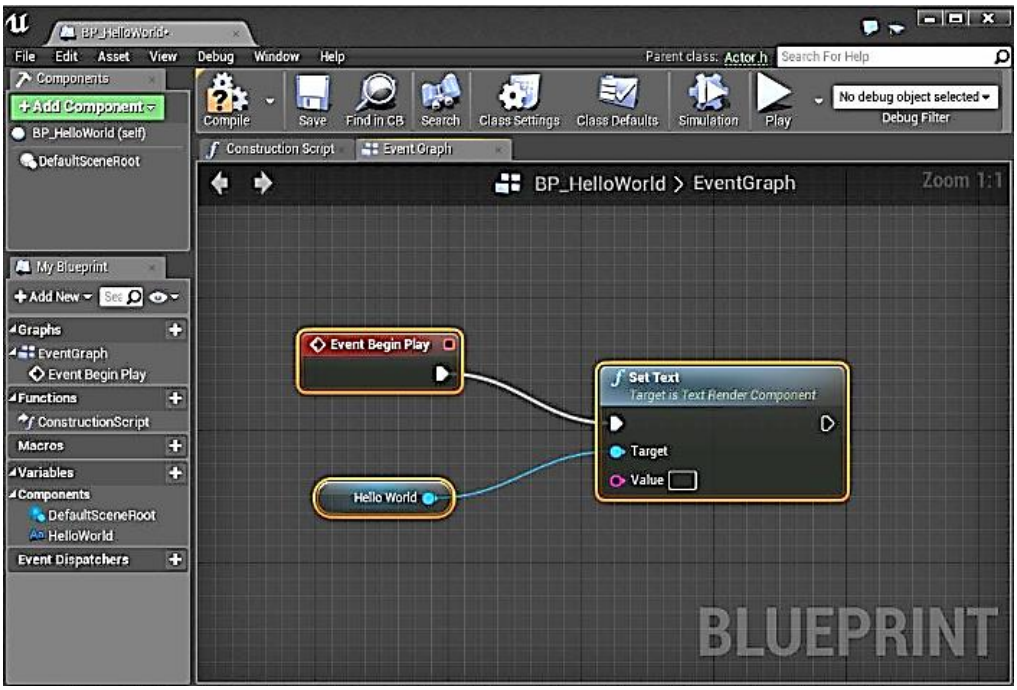


Рисунок 2.2 - Скриптинг через візуальну систему Blueprint

Переваги:

- гарна оптимізація для сучасних пк та ігрових приставок;
- великий обсяг документації та активна підтримка розробників;
- одна з найкращих систем обробки зображень у сфері мультимедійних механізмів;
- наявність системи візуальних сценаріїв blueprint за замовчуванням;

- безкоштовне використання повної версії, доки ви не заробите 3000 доларів сша на квартал, потім 5% скорочення від розробників двигуна;
- відкритий код двигуна;

Недоліки:

- передова розробка продуктів для мобільних ос (android, ios);
- інтерфейс середньої складності використання;

2) Unity

Цей двигун відомий своєю здатністю розробляти програми для різних платформ, а також є можливість створювати веб-додатки. Unity користується великим попитом серед розробників ігор, особливо для мобільних платформ. Rishi виділяється серед інших частими оновленнями, добре продуманим інтерфейсом редактора (рис. 2.3) і підтримкою готових рішень для комерційної складової в своїх додатках, тобто готових рішень у вигляді різних плагінів для розробки [4]. Реклама, аналітика, внутрішні покупки, ігровий центр і т.д. Це причина, чому Unity набула популярності, особливо серед розробників мобільних додатків.

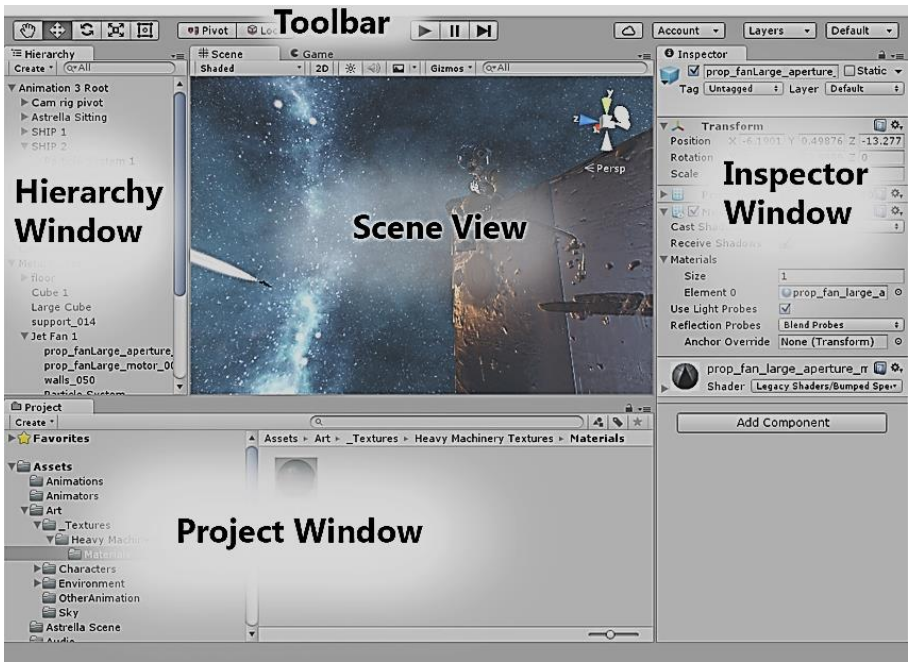


Рисунок 2.3 - Інтерфейс редактора Unity

Що стосується графіки, то цей движок задовольняє потреби користувача в приємному оку зображенні, але він не підійде для проектів з кінематографічною графікою. З іншого боку, якщо ми згадаємо можливість націлювання на портативні пристрої, тобто

Здебільшого існують не надто потужні пристрої на базі ОС Android або iOS, і, враховуючи реальний розмір екрану, така велика деталізація зображення може бути перевагою. А також розглядають можливість створення повноцінної 2D-графіки. Якщо використовувати движок, то його можливості рендеринга зображень загалом еквівалентні тому ж Unreal Engine 4 [8].

Що стосується можливих способів створення програмного коду, то підтримуються мови C++, C, JavaScript і деякі інші. Але якщо вам потрібно працювати з движком на професійному рівні, ви повинні добре володіти мовою C#.

Є аналог Blueprint під назвою PlayMaker for Unity, але на відміну від першого, Playmaker поширюється як розширення для редактора, та ще й на платній основі (рис. 2.4).

Рисунок 2.4 - Візуальна візуалізація сценаріїв за допомогою розширення PlayMaker для Unity

Сам драйвер має умовно безкоштовну модель розповсюдження: якщо користувач заробляє менше $100 000 на рік від продажу програмних продуктів, побудованих на поточному драйвері, то користувач має повне право безкоштовно використовувати базову (неповну) версію Unity. . В

іншому випадку вам потрібно одноразово придбати повну версію за 1500 доларів або скористатися підпискою, яка коштує 75 доларів на місяць.

Незважаючи на очевидні переваги, є досить неприємний недолік, з яким сходяться практично всі користувачі даного движка - відсутність своєчасного усунення помилок, що виникають при оновленні. Таким чином, якщо проблема розробки того чи іншого додатка стосується саме інструментарію Unity, то необхідно або можливо переписати створений користувачем код.

Впливає на проблемні функції двигуна або навіть зупиняє розробку, доки розробники не виправлять відповідні помилки в наступному оновленні.

Переваги:

- велика кількість довідкових джерел для розробки на поточному двигуні;
- керівництво по створенню мобільного додатку;
- легке вбудовування рішень, розроблених для монетизації проекту;
- велика кількість цільових платформ розробки;
- неперевантаження основного інтерфейсу;

Недоліки:

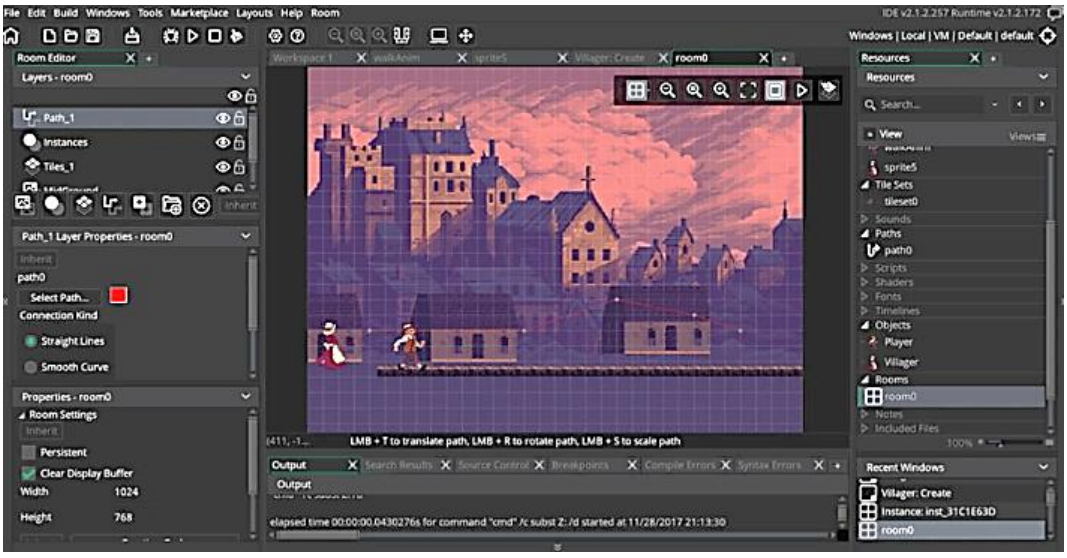
– базова версія якісно недопрацьована;
- хоч регулярні, але інколи проблематичні оновлення;
- відсутність візуальних сценаріїв за замовчуванням;
- закриття вихідного коду двигуна;

3) Game Maker Studio 2

Rishi призначений для користувачів, які не мають можливості або бажання вивчати складні мови програмування [17]. Тобто сам движок максимально простий, коли створена логічна та графічна частина програми. Часто для взаємодії з інтерфейсом тут використовується метод взаємодії drag-and-drop (від англійського «drag-and-drop», по суті, це перетягування графічних елементів, наприклад, комп'ютерною мишею). Таким чином, робота з Gamemaker Studio 2 є винятковою. З усіх розглянутих аналогів цей движок має найпростіший для розуміння інтерфейс редагування (рис. 2.5),

що забезпечить швидке використання в процесі створення програми. Крім того, є не тільки стандартні засоби розробки (графіка, звук, мережеве спілкування), але і вбудований графічний редактор, який може працювати, наприклад, зі спрайтами. Спрайт - це зображення, яке є окремим графічним об'єктом у програмі.

Також варто відзначити, що поточний движок працює тільки з 2D-графікою, хоча є можливість працювати і з 3D. У той же час слід зазначити, що Gamemaker Studio 2 все ще є движком 2D-екшн, тому процес розробки тривимірної графіки тут дуже непопулярний і погано визначений, а зображення при обробці тривимірних об'єктів це не так. здається робити. Дуже добре, але процес візуалізації (використання комп’ютерної програми для отримання зображення на моделі) не кращий.



Риснок 2.5 - Інтерфейс редактора Gamemaker Studio 2

Він також має власну мову програмування GML (GameMaker Language) для створення програм на основі цього двигуна. У цій мові сценаріїв використовується інтерпретатор (програма, яка обробляє код для кожного рядка чи команди та негайно виконує його; компілятор, наприклад, виконує код лише після його повної обробки.), синтаксис подібний до C, C++ і JavaScript. . Розробники відзначили, що хоча GML і є мовою програмування, вона набагато простіша за вищезгадані C або C++.

Також цікаво, що оскільки GML є мовою сценаріїв, вона відповідає всім вимогам візуального сценарію завдяки численним додаткам і зручному інтерфейсу редактора, який не завжди доступний у тому самому механізмі, але це дуже зручний інструмент. Зробити код не тільки в текстовому редакторі, але і через взаємодію візуальних блоків (рис. 2.6).

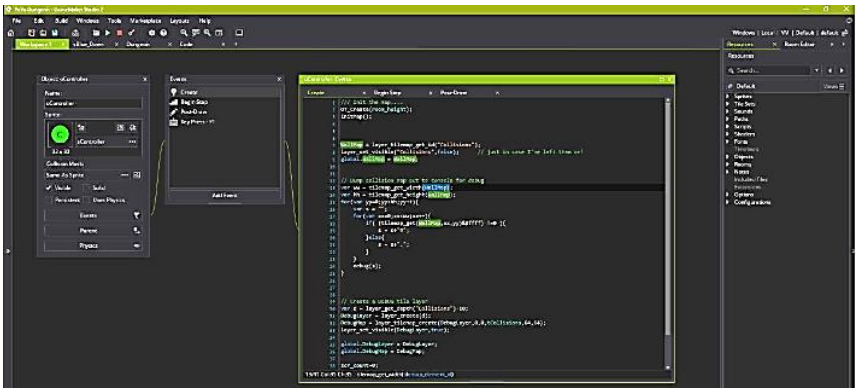


Рисунок 2.6 - Візуальний сценарій у Gamemaker Studio 2

Однією з особливостей движка є те, що вся графіка для майбутнього додатку завантажується у відеопам'ять у вигляді атласу - великої картинки, куди вставляються всі графічні елементи для відображення у 2D. Таким чином економляться ресурси системи, яка завантажує всю графіку один раз і не «відкриває» ресурси заново під час роботи програми. Крім того, атласи можна повністю конфігурувати - це може бути корисно для поділу атласу на кілька частин для поступового або змінного завантаження у відеопам'ять або його можна використовувати під час портування програми між різними платформами [6]. Щоб перемістити: Наприклад, якщо ви потрібно рухатися. Програма з великими екранами (ПК, ігрові консолі) з маленькими екранами (мобільні пристрої), тоді вона здатна зменшити роздільну здатність атласу, який вже буде створюватися на системах з меншою кількістю відеопам’яті та меншими розмірами екрану.

*Game Maker Studio 2*Розділяється на три версії: стандартна (безкоштовна, неповний набір інструментів для розробки, найгірша – відсутня можливість перенесення проектів на більшість систем – втрачається

кросплатформна функціональність, працює лише в системах Windows), професійна (ціна 100 доларів, включає повний набір інструментів). , кросплатформенність - можливість роботи під macOS, *Ubuntu*і і Android), а також Master Collection (ціна 800 доларів США, кросплатформність і доступність для розробки всіх додаткових модулів).

Переваги:

- Дуже простий і простий інтерфейс редактора;

-Наявність оптимізації під 2D-графіку;

- Активна спільнота користувачів, які завжди готові допомогти;

- Велика кількість інформації, доступної для використання водієм;

- Власна мова програмування, яка дозволяє легко розробляти програми в рамках основних можливостей двигуна;

- Оскільки вбудована мова сценаріїв, можливість візуального сценарію доступна відразу;

Недоліки:

- Безкоштовна версія движка дуже «низька» за можливостями

- Майже повна відсутність зовнішніх бібліотек для розширення інструментарію;

- Шрифт дуже складний для роботи (масштабування, дизайн тощо);

- Підтримка постобробки зображень, хоча й присутня, не працює належним чином;

4) Cocos2D-x

Оригінальний двигун Cocos2D був розроблений у 2008 році на мові програмування Python, потім портований для розробки iPhone (на мові Objective C). Через два роки була випущена перша версія Cocos2D для кросплатформенної розробки, яка підтримувала мову C++. Ця версія отримала назву Cocos2D-x.

В даний час ця версія двигуна розробляється на мовах програмування Java, JavaScript, Lua і C# для створення додатків для різних ОС. Cocos також є розширенням для візуалізації 3D-графіки, але, як і у випадку з рушієм, що

обговорювався раніше, можливості такої програми є досить розширеними, оскільки Cocos2D зосереджується на двовимірній графіці [7].

Цей движок дуже популярний не лише серед любителів та незалежних розробників, а й серед таких гігантів галузі, як Google, Microsoft та Intel. В основному це пов’язано з тим, що Cocos2D-x зберігає «золоту середину»: перш за все, це добре продумана мультимедійна бібліотека, яка забезпечує функціональність.

Графіка, звук або пам'ять, створені на рівні компонентів під час написання коду. Але крім цього для цього движка створено спеціальний редактор Cocos Creator (рис. 2.7), який містить набір необхідних інструментів для анімації, управління ресурсами тощо. Цей редактор офіційно рекомендований до використання виробниками двигунів. При цьому використання редакторів, відладчиків, компіляторів та інших інструментів, наданих розробниками, які зазвичай застосовуються при установці драйверів, не є обов'язковим. Як середовище можна використовувати, наприклад, Visual Studio з *Microsoft*і підключіть Cocos2D-x як додаткову бібліотеку до проекту [9].

Рисунок 2.7 - Попередній перегляд інтерфейсу Cocos Creator

Така гнучкість використання, безкоштовна і відкрита модель розповсюдження і можливість опціонального підключення офіційних інструментів, таких як «Військовий» редактор з графічним інтерфейсом для

движка і Cocos2D-x відрізняється не лише від інших 2D-движків, але й від програмного забезпечення для розробки мультимедійних додатків загалом.

Переваги:

- Безкоштовна ліцензія на використання та розповсюдження (MIT License);
- Відкритий вихідний код;
- Можливість використання в інших середовищах програмування;
- Наявність офіційного програмного редактора від розробників двигуна;
- Популярність серед гігантів індустрії, як результат перспективи;
- Відносно велика кількість мов, доступних для програмування двигуна.

Недоліки:

- Відсутність вбудованої передачі планів в ігрові консолі;
- Відносно невелика аудиторія користувачів двигуна;
- Деякі офіційні додаткові засоби розробки та контент;
- Незвичайні оновлення двигуна.

2.2. Аналіз умов аерофотозйомки

Камери БПЛА та їх характеристики

Камера є одним з основних компонентів дрона, і багато людей вирішують, який квадрокоптер купити, виходячи виключно з параметрів камери. Тому, якщо призначення дрона не обмежується дитячими пожертвами, велику увагу варто приділити вибору камери. Перш за все, необхідно визначитися, який тип камери вам потрібен: вбудована або виносна.

Вбудована камера.

Весь апарат камери розташований на корпусі дрона [21]. Як правило, вбудована камера має обмежену мобільність, але сам дрон більш маневрений. Такі камери можуть бути як бюджетними (їхньої якості якраз вистачить для фотографій), так і професійного рівня з можливістю зйомки відео в 4K. Топові

моделі квадрокоптерів оснащені високоякісними камерами у співпраці з відомими брендами (Hasselblad, Leica).

Дистанційна камера.

Дрони з дистанційними камерами мають характерну конструкцію, яка дозволяє прикріпити професійну фото-, відеокамеру чи екшн-камеру. Переваги такого дрона очевидні, ви можете замінити всю камеру на власний пристрій і отримати звичну якість зйомки [10]. У продажу також є безкамерні безпілотники, це заощадить бюджет.

Проте з точки зору параметрів камери вибір показників на дроні мало чим відрізняється від вибору показників на смартфоні. Чим дорожче пристрій, тим краще буде камера на борту.

Ось кілька порад новачкам, щоб вибрати квадрокоптер з камерою:

Діапазон роздільної здатності в камері може бути від 0,3 Мп до 4K. Це впливає не тільки на якість зображення, а й на форму кінцевого кадру. Якщо ваша мета - фото в соціальних мережах, не варто гнатися за високою роздільною здатністю. 720p і вище вважається хорошим показником для новачків. Навіть Full HD (1080p) може бути недостатньо для професійної зйомки відео, варто придивитися до квадрокоптерів з камерами 4K (2160p).

Якщо виділити особливо важливі параметри для таких камер, то це кут огляду (хорошим вважається не менше 80 градусів), якість підвіски і стабілізація, оскільки це особливо важливо для отримання стабільного зображення в польоті.

2.2.2. Аерофотозйомка.

Аерофотозйомка є одним із найефективніших методів отримання просторових даних. Висока візуальна ізоляція та якість фотографій забезпечують застосування широкого спектру отриманих даних у різних сферах діяльності.

Аерофотозйомка — це зйомка місцевості за допомогою аерофотоапарату (АФА), встановленого на літаку або гелікоптері.

Аерофотозйомка (АФЗ) - це фотозйомка частин земної поверхні з літака для створення топографічних карт або виконання народногосподарських завдань.

Аероілюстрація залежності α головної оптичної осі АФА від кутів відхилення від вертикалі (рисунок 2.8).

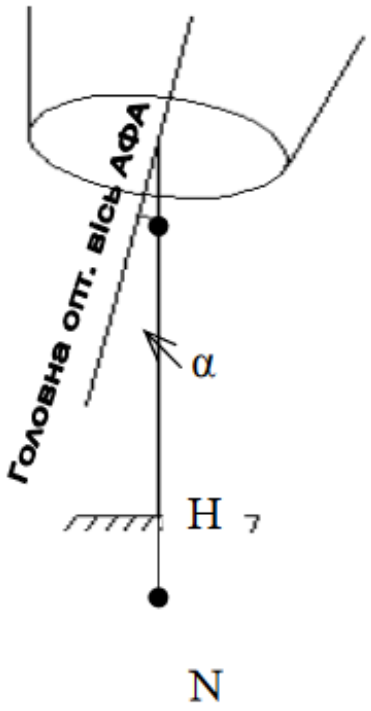


Рисунок 2.8 – Відхилення головної оптичної осі АФА від вертикальної лінії

Залежно від кутів α плоску діаграму поділяють на:

- площина (кут $\alpha \leq 30$);
- Перспективний (кут $\alpha > 30$).

Планова зйомка ведеться з використанням гіростабілізації аерофотоапаратів, що дозволяє значно зменшити кути розбіжності (α = 10-15', α = 45-60').

Для розв'язування топографічних задач застосовуються тільки використовувані АФЗ [11]. Залежно від поставленого завдання і розміру місцевості, а також кількості і розташування використовуваних аерофотознімків аерофотозйомка поділяється на:

- кадри (одиночні), які забезпечують одне або кілька одиничних зображень невеликих ділянок місцевості;

− Дорога – для фотозйомки лінійних об’єктів (дороги, річки, берегові лінії), які забезпечують зображення з накладеними зображеннями смуги місцевості поблизу лінії польоту.

- Multi-path (область) - для двох або більше паралельних шляхів для фотографування області, які перекриваються.

Залежно від масштабу зйомки топографічна аерофотозйомка може бути:

- дрібномасштабні - 1:500 000 і дрібніше;

- Середній масштаб - 1:500000 - 1:100000;

- Великомасштабні - 1:100000 і так далі.

Ділянки земної поверхні фотографуються в межах трапеції топографічних карт. Аерофотозйомка ведеться, як правило, прямими паралельними курсами із заходу на схід і навпаки, північ або південь.

Дороги повинні бути безперервними та паралельними межам ділянки зйомки, крайні осі доріг збігатися з межами ділянки. Основні технічні вимоги до аерофотозйомки: Мала нелінійність доріг, що характеризується відношенням прогину осі дороги до її довжини, а 2 при зйомці в масштабах менше ніж 1:5000 % не повинно перевищувати висот понад 750 м, а також 3 % при зйомці на висотах нижче 750 м у масштабі 1:5000 і більше.

Аерофотознімки місцевості Це роботи, які базуються на ортофотопланах та фотопланах місцевості. Аерофотозйомка місцевості дозволяє робити знімки на потрібній висоті і в потрібний час у заданому місці. Якісна аерофотозйомка ділянки землі відрізняється високою точністю і багатьма перевагами.

Області застосування аерофотозйомки

1) Оперативне картографування

Регіон наносять на карту за допомогою літаків і безпілотників. Топографічне картографування земель можна виконувати з різних висот. Послуги аерофотозйомки дуже актуальні в Україні, адже аерофотозйомка ділянки допоможе вчасно помітити будь-які зміни в зовнішньому середовищі та спрогнозувати подальший розвиток подій ГІС-картографія більш відома,

включає створення та використання електронних та цифрових карт. на основі ГІС-технології. Оперативне картографування активно використовується для оновлення карт і виготовлення карт рельєфу і рельєфу. Картографування місцевості є важливим для регіонального планування, і такі послуги дозволяють проводити просторово-часовий аналіз.

Особливістю технології оперативного картографування є мобільна зйомка заданої території за допомогою безпілотних літальних апаратів. Через певні проміжки часу апарат робить знімки місцевості відповідно до заданої висоти під час польоту, моделі камери, накладання знімків та інших характеристик, а її координати додаються до кожної фотографії з точністю до сантиметра. Продукт аерофотозйомки сумісний із більшістю пакетів прикладного програмного забезпечення для створення ортофотокарт і хмар точок.

Оперативні карти призначені для вирішення широкого кола завдань [12]. Перш за все, слід розрізняти два типи оперативних карт для попереджень і сигналів про несприятливі або небезпечні процеси, для спостереження за їх розвитком, складання рекомендацій і прогнозів, вибору варіантів управління, стабілізації або зміни ходу процесів у різних сферах — від екологічних умов до політичних. події: одні призначені для довгострокового використання та аналізу (наприклад, карти, які надають огляд статистики чи інших характеристик окремих регіонів), тоді як інші призначені для короткострокового використання для негайної оцінки будь-якої ситуації (наприклад, карти зрілості). сільськогосподарських культур).культур).

Основні переваги оперативного картографування:

- динаміка процесу;
- безпека;
- економічність праці;
- висока швидкість;
- геодезична точність;
- простота використання;
- Інформативність даних.

Оперативне картографування вирішує задачі оперативного картографування, обстеження сільськогосподарських угідь, моніторинг втрат тепла в будівлях і трубопроводах, вимірювання обсягів сипучих матеріалів, визначення топографічних моделей місцевості

2) Підвищення ефективності сільського господарства.

Важливою галуззю є сільське господарство. А показником рентабельності сільськогосподарського виробництва є врожайність, яка є істотною для контролю. Прогнозування врожайності прогнозує якість і кількість врожаю, а також дозволяє спланувати комплекс робіт для максимізації врожаю.

Аерофотозйомка полів дозволяє на ранній стадії прорахувати кроки, що інакше зробити важко. У сільському господарстві аерофотозйомка допомагає спостерігати та виявляти різні проблемні ділянки полів, забезпечуючи оцінку величини впливу, а також джерела виникнення таких ділянок. Аерофотозйомка для точного землеробства може ефективно каталогізувати землі під посіви.

Сільськогосподарський моніторинг може вивчати зміни ґрунтів і картувати їх зміни [13]. На фермі аерофотозйомкою визначають ділянки з водою, ділянки з ерозією ґрунту, а також сухість або зволоженість ділянки. Аерофотозйомка посівів дає детальну інформацію про стан посівів, яку важливо знати фермерам.

Порівняно з супутниковим і повітряним відстеженням безпілотні багатороторні системи мають принципові переваги не тільки з точки зору точності відстеження та вартості, але й з точки зору їх можливості використання в умовах низької хмарності. Використовуючи сучасні технології, такі як ідентифікація та обчислення, прогнозування очікуваний розмір врожаю, контроль якості польових робіт, виконаних сільськогосподарською технікою, ми з точністю до 20 мм на піксель. Можемо зробити детальну карту ділянки.

3) Повітряний малюнок у сфері архітектури.

Аерофотозйомка в архітектурі може представити як готові будівлі, так і нові проекти. Це ефективний спосіб представити поточний стан розвитку або

розвитку об'єкта його власнику або архітектору. Аерофотозйомка нерухомості дозволяє ріелторам швидко здійснити продаж і створити позитивне враження про клієнта або покупця.

Архітектурна аерофотозйомка дозволяє побачити куточки місцевості, недоступні для огляду з поверхні землі. Тобто ви можете побачити помилки та важливі функції, які допоможуть у майбутньому. У сфері архітектури аерофотозйомка забезпечує реальну 3D візуалізацію.

Аерофотознімки пам'яток архітектури дають найбільш точну інформацію про об'єкт і ділянку землі. Але з ним можна знімати з будь-якого кута і з великої висоти. Аерофотозйомка також популярна для реконструкції архітектурних споруд, адже без неї важко оцінити фактичний стан будівлі та скласти план дій.

Для виконання завдань, пов'язаних з реконструкцією архітектурних споруд, необхідно мати точну модель споруди. У більшості випадків архітектор не має повного комплекту креслень фасаду та елементів декору. У цій ситуації вихід – використання симбіозу технологій лазерного сканування та аерофотозйомки. Цей метод дозволяє сфотографувати видиму поверхню будівлі з усіх можливих ракурсів.

Основні переваги використання аерофотозйомки в сфері архітектури:

- геодезична точність отриманої моделі;
- швидкість фільму;
- інформативність;
- безпека;
- мобільність;
- економіка.

4) Розрахунок обсягу.

Одним із напрямів інженерних вишукувань є кількісний розрахунок, обстеження та оцінка вартості сипучих матеріалів. Класичні методи дають рішення з високими похибками, а сам процес калібрування займає багато часу. Сьогодні за допомогою аерофотозйомки можна швидко та безпечно оглянути потрібну територію, а за допомогою постобробки отримати об’єми з похибкою

до 1%. Одна тільки простота процесу дозволяє робити це з необхідною періодичністю.

Аерофотозйомка для розрахунку об'єму - це послуга, яка користується великим попитом в будівництві, гірничодобувній промисловості, промисловості та сільському господарстві [15]. Аерофотозйомка для розрахунку об'єму потрібна для багатьох земляних робіт, а також для транспортування сипучих матеріалів. Розрахувавши обсяг сипучих матеріалів, можна відстежувати їх і проводити аудит закритих і відкритих складів.

Кількість землі можна підрахувати за допомогою дрона. Щоб виправити аерофотозйомку для розрахунку об’єму, необхідно провести топографічну зйомку, яка покаже всі нерівності та особливості рельєфу, тому вимірювання об’єму сипучих матеріалів є відповідальним завданням, яке повинні виконувати лише досвідчені фахівці.

У геології послуга розрахунку кількості земляних робіт при вертикальному плануванні визнана вигідним рішенням для забудовника, оскільки позбавляє від зайвих витрат на кожному етапі: від кошторису на закупівлю будматеріалів, оформлення договорів оренди до спеціальних інструментів для зведення фундаменту. Інженерно-геодезичні дослідження – єдина можливість контролювати та визначати бюджет:

- точні оцінки;
- доцільність перенесення ґрунту (на прилеглу територію або до плану вивезення);
- прогрес у реалізації проекту;
- правильність ведення обліку зберігання матеріалів;
- геодезична точність;
- точна кількість матеріалу, що експортується;

Сучасне обладнання та програмне забезпечення дозволяють точно та швидко обробляти матеріали. Топографічна аерофотозйомка повністю відповідає всім геодезичним вимогам щодо точності. Аерофотозйомка проводиться в три етапи: підготовчий етап, польові роботи та операторська робота.

Послуги аерофотозйомки в Україні актуальні, так як аерофотозйомка ділянки допоможе помітити будь-які зміни зовнішнього середовища з часом і передбачити розвиток подій в майбутньому

2.3. Висновки за розділом

В другому розділі були проаналізовані програмні аналоги візуальної частини модуля керування БПЛА, тобто мультимедійні движки, призначені для створення 3D або 2D графіки певним чином і відповідно до наданих даних.

Також в цьому розділі розглянуто та проаналізовано умов аерофотозйомки.

3 ОПИС МАТЕМАТИЧНОЇ МОДЕЛІ

3.1. Кінематичне проектування моделі руху БпЛА

Вагомий внесок у становлення та розвиток кінематичного прогнозування вчених Львівської політехніки В. М. Гелаговського та І. Г. Пулкевича. У своїх працях [4; 5] першим запропонував використання лінійних операторів для представлення грамографії, ротографії та спінографії рухомих об’єктів у просторі. Поряд із розробкою алгоритмів розв’язування прямої задачі кінематичного прогнозування для знаходження прогнозів траєкторії просторового переміщення об’єктів були розроблені та ретельно досліджені алгоритми розв’язування оберненої задачі. Зворотна задача полягає в знаходженні координат просторового розташування об'єкта по заданому шляху. Дослідження вчених-геометристів на сучасному етапі розвитку прикладної геометрії та інженерної графіки в галузі вдосконалення методів і техніки проекційного відображення в цілому, і так званого динамічного, або кінематичного проектування, зокрема, можна умовно розділити. За кількома основними напрямками, перш за все, це робив професор НТУ «Київський політехнічний інститут імені Ігоря Сікорського» Ванін В.В. для вдосконалення методології практичного застосування відомих методів проектування та створення та дослідження особливостей новітнього способи відображення елементів простору.

Порівняно зі способом визначення координат БПЛА, який базується на дистанційно-різницевому радіолокаційному методі пошуку положення БПЛА в повітряному просторі, та іншими методами радіолокаційного пошуку, запропонований спосіб визначення координат БПЛА усунений кінематикою. У них є ряд недоліків, зокрема, насамперед, нечутливість до перешкод, викликаних зворотними радіохвилями, які відбиваються від нерівностей рельєфу земної поверхні, висотних будівель та інших природних аномалій. Це пов'язано з тим, що на відміну від відомих методів

радіолокаційного визначення координат рухомих об'єктів, в яких відстань до літака і, відповідно, його координати, розраховуються шляхом реєстрації часу проходження відбитих радіохвиль. У пропонованому способі радіохвилі, відбиті від літака, реєструються тільки для визначення напрямку розташування об'єкта. Суть запропонованого способу наведена на рис. 3.1-3.6.

На рисунку 1 показано розміщення на землі радіолокаційних станцій *A*≡Радар №1 і *B*≡Радар №2, які віддалені одна від одної на відстань 500-1000 метрів. Точка *C*≡КП вказує на умовне розташування командного пункту (КП), який знаходиться на певній відстані від обох РЛС. Три точки A, B і C визначають базову площину відліку, позначену α (A, B, C).

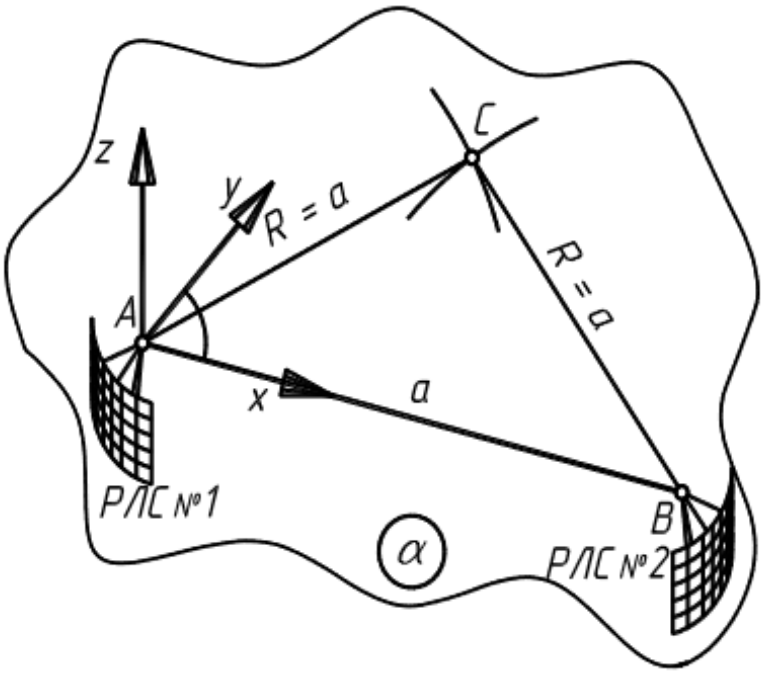


Рисунок 3.1 - Базова площина α відліку із декартовою системою координат

В одному з радарів, наприклад, в точці A, вводиться тривимірна декартова система координат з перпендикулярними одна до одної осями x, y і z. Вісь X цієї системи координат починається в точці A і йде в напрямку точки B, яка є умовним позначенням місця і координат установки другого

радара. Вісь z починається з точки А, перпендикулярна до осі х і спрямована вгору.

Вісь у теж започатковується в точці *A* і перпендикулярна осям *x* та *z*. Осі *x* та *y*, як дві взаємно перпендикулярні прямі, утворюють так звану “базову” площину α. Саме у цій базовій площині α і облаштована проміжна точка *C*, що рівновіддалена на віддаль *a* від обох РЛС.

Задавши висоту *H*, що перевищує в 1,2–1,5 рази орієнтовну висоту польоту БПЛА, на перпендикулярах до базової площини α в точках *A, B* та *C* встановлюють точки *S, W* та *L*. Ці три точки *S, W* та *L* задають у просторі “картинну” площину β(*S;W;L*), яка паралельна базовій площині α(*A;B;C*) і віддалена від неї на віддаль *H*, тобто *H=AS=CW=BL*; α(*ABC)*||β*(SWL*) (рис. 3.2).

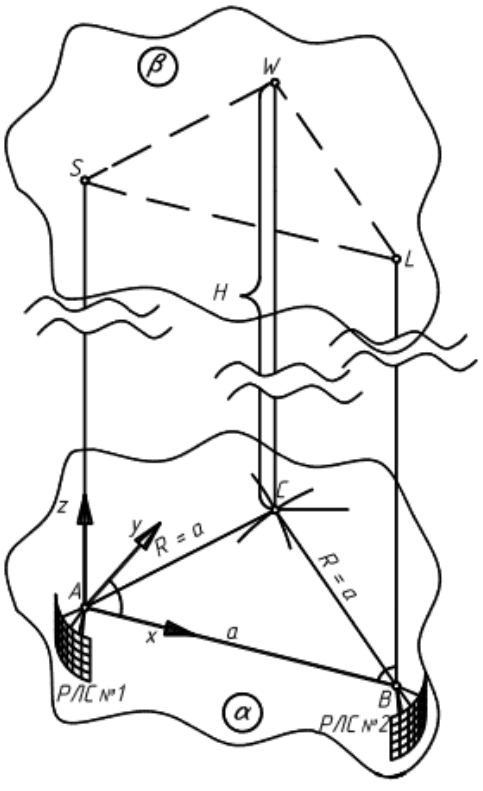


Рисунок 3.2 - Задання “картинної” площини β розрахункової схеми

Увімкнувши обидві РЛС, спрямовують в очікуваному напрямку розташування БПЛА електромагнітні радіохвилі (рис. 3.3). На моніторах РЛС фіксують напрям проектуючих променів, що проходять від кожного із

РЛС через точку просторового розташування БПЛА, та кути їх нахилу до базової площини α, тобто $\gamma = p_1 \hat{} \alpha$ та $\sigma = p_2 \hat{} \alpha$. Крім того, для повноцінної координатної прив'язки проектуючих променів до запровадженої системи координат для кожного із проектуючих променів p_1 та p_2 визначають і кут його нахилу до лінії a, що з'єднує між собою обидві РЛС.

Тобто $\delta = p_1 \hat{} AB$, $\phi = p_2 \hat{} AB$ (рис. 3.3).

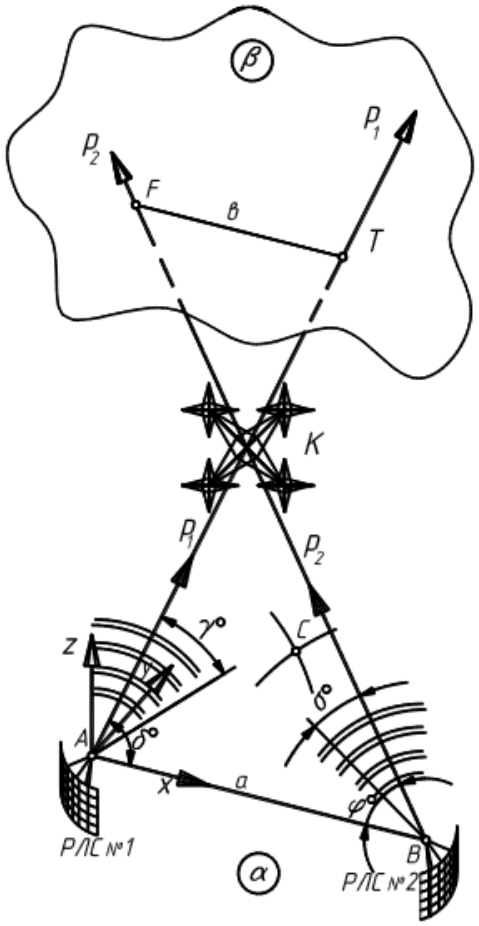


Рисунок 3.3 - Схема спрямування проектуючих променів p_1 та p_2

За координатами точок розташування РЛС і кутами γ, σ, δ, ϕ нахилу проеціюючих променів до основної площини α і прямої a, що з'єднує обидва РЛС, ЕОМ командного пункту відображає обидва зовнішні промені p_1 та p_2.

За відповідною комп'ютерною програмою знаходять координати точок $F = p_2 \cap \beta$ та $T = p_1 \cap \beta$ перетину проектуючих променів p_1 і p_2 із картинною площиною β.

Визначають відстань b між точками перетину проектуючих променів із "картинною" площиною β, тобто b=|FT|.

Точку перетину проектованого променя з площиною «зображення» β шукають аналітично або графічно.

3.2. Аналітичний метод знаходження координат БпЛА

В аналітичному методі знаходження точки перетину аналітичне рівняння променя, що проектує, задається як рівняння прямої, яка проходить через відому точку (А або В) із заданими координатами під певним кутом нахилу до " база». площину α, та аналітичне рівняння «зображувальної» площини β, яка проходить через три точки S, W і L із заданими координатами.

У цьому випадку точкою перетину прямої з площиною буде точка, координати якої одночасно задовольняють і рівняння прямої, і рівняння площини. При використанні графічного методу координати точки перетину опуклого променя (p_1 або p_2) із “картинною” площиною β шукають за методикою пошуку точки перетину прямої з площиною. Тобто через проектуючий промінь p_1 або p_2 проводять додаткову площину посередник μ, яка перпендикулярна “картинній” площині β, а саме: $p_1 \subset \mu_1 \perp \beta$, $p_2 \subset \mu_2 \perp \beta$. Шукають лінію l перетину площини-посередника μ із “картинною” площиною β, тобто $l_1 = \mu_1 \cap \beta$, $l_2 = \mu_2 \cap \beta$. Де знайдена лінія l перетинається із проектуючим променем p там і буде шукана точка, зокрема:

$$T = p_1 \cap l_1 = \mu_1 \cap \beta\,,\ F = p_2 \cap l_2 = \mu_2 \cap \beta\,.$$

З двох подібних трикутників ΔАВК і ΔFTK, які утворені променями, що перетинаються, відраховуються координати їх спільної вершини в точці К. Цією точкою є К, де зараз знаходиться БПЛА (рис. 3.4). Координати точки К визначаються, наприклад, як координати точки, розташованої на промені пускової установки р1 або р2, що знаходиться на певній відстані t від одного з РЛС.

Відстань t визначають із подібності утворених пересічними проектуючими променями p_1 та p_2 подібних трикутників ΔAKB та ΔFKT.

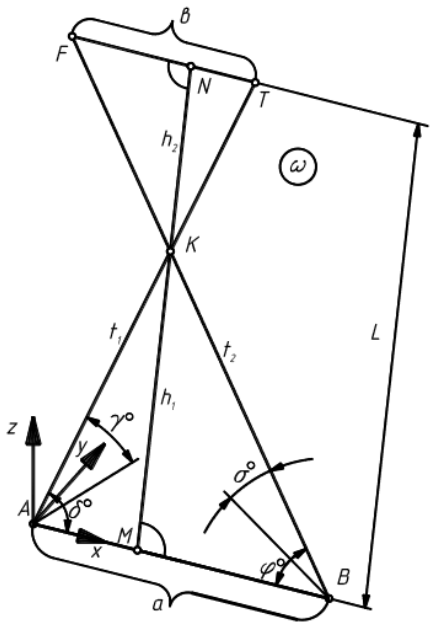


Рисунок 3.4 - Розрахункова схема визначення віддаленості БПЛА від радіолокаційних станцій

Сума висоти h_1 та h_2 цих трикутників, що проведені із спільної вершини K, рівна

$$h_1 + h_2 = L = \frac{H}{\sin \nu},$$

де ν – кут між площиною ω, утвореною перетином виступаючих балок p1 і p2, і площиною основи α, утвореною осями x і y (рис. 3.5).

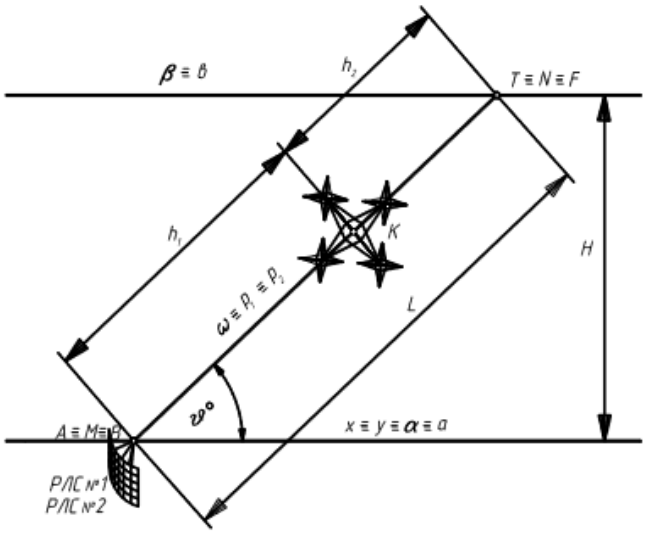


Рисунок 3.5 - Просторове розташування БПЛА між базовою α та “картинною” β площинами

Із подібності трикутників ΔAKB та ΔFKT отримуємо:

$$\frac{h_1}{AB} = \frac{h_2}{FT};\ h_1 + h_2 = L.$$

Тоді

$$h_1 = L - \frac{bL}{a+b} = L\left(1 - \frac{b}{a+b}\right);\ h_2 = \frac{bL}{a+b},$$

де a та b – основи трикутників, протилежні їх спільній вершині K**;**

$$L = h_1 + h_2 = \frac{H}{\sin\nu}$$

– відстань між основами a та b подібних трикутників;

h_1 – висота трикутника ΔAKB;

h_2 – висота трикутника ΔFKT;

H – відстань між базовою α та “картинною” β площинами.

Отже, із прямокутного трикутника ΔAKB

$$t_1 = AK = \sqrt{(AM)^2 + (KM)^2} = h_1\sqrt{1 + (\operatorname{ctg}\delta)^2}.$$

Із прямокутного трикутника ΔBKM виходить:

$$t_2 = BK = \sqrt{(BM)^2 + (KM)^2} = h_1\sqrt{1 + (\operatorname{ctg}\phi)^2},$$

де δ_1 та ϕ_1 – відповідно кути нахилу проектуючих променів до осі x запровадженої системи координат;

t_1=АК – відстань по довжині проектуючого променя p_1 від РЛС №1 до БПЛА;

t_2 =ВК – відстань по довжині проектуючого променя p_2 від РЛС №2 до шуканого БПЛА.

Володіючи відстанями t_1 і t_2 від РЛС №1 і РЛС №2 до точки K на проектуючих променях p_1 і p_2, тобто до точки, у якій на даний момент часу знаходиться шуканий БПЛА, на комп’ютері командного пункту перепроектовують знайдену точку K на координатній осі запровадженої системи координат. Знайдені значення на відповідних осях x, y та z і будуть шуканими координатами БПЛА (рис. 3.6).

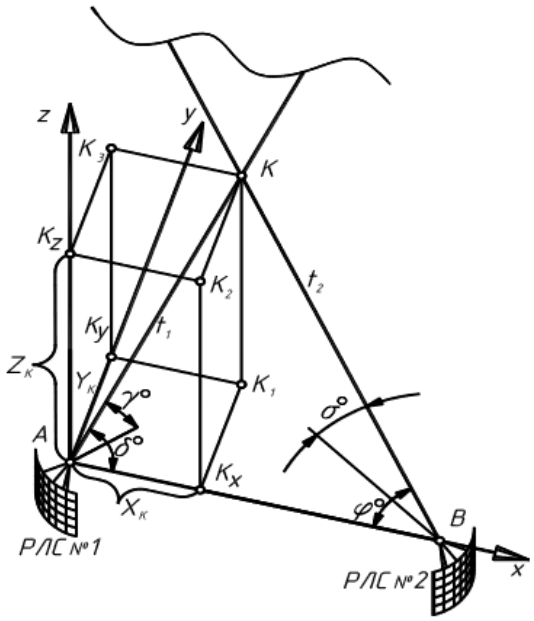


Рисунок 3.6 - Визначення координат БПЛА ортогональним проектуванням

А саме:

$$x_K = K_x A = t_1 \cos\delta\,;$$
$$y_K = K_Y A = h_1 \sin\nu\,,$$

де δ – кут нахилу проектуючого променя p_1 до лінії a=|AB|, що з'єднує між собою обидві РЛС;

ν – кут нахилу утвореної проектуючими променями p_1 та p_2 площини до базової площини α.

Таким чином, у введеній декартовій системі координат з початком її осей у точці А, яка є символом розташування одного з радарів, можна без двозначності показати місцезнаходження потрібного дрона [19]. Це можна зробити шляхом значних розрахунків довжин відрізків t_1=AK та t_2=BK, розташованих на проеціюючих променях p_1 та p_2, і кутів нахилу до базової площини, відомих з моніторів радіолокацій. $\delta = \omega(p_1 \cap p_2)\,\hat{}\,\alpha(ABC)$ Площина ω, утворена проектуючими променями, а також відомі кути нахилу δ і γ проектуючих променів p_1 та p_2 до лінії, що з'єднує обидва радари.

Таблиця 3.1 - Розраховані залежності для визначення координат БПЛА

№ з/п	Розрахункові параметри: Назва параметра	Позначення	Математичні залежності розрахунку параметрів	Значення параметрів при куті ν нахилу площини проектуючих променів $\omega(p_1 \cap p_2)$ до базової площини $\alpha(\Delta ABC)$ $\nu = \omega(p_1 \cap p_2) \wedge \alpha(\Delta ABC)$		
1	Кути нахилу: а) площини проектуючих променів до базової площини	ν	$\nu = \omega(p_1 \cap p_2) \wedge \alpha(\Delta ABC)$	60°	70°	80°
	б) першого проектуючого променя p_1 до лінії між РЛС	δ	$\delta = p_1 \wedge a$	70°		
	в) другого проектуючого променя p_2 до лінії між РЛС	ϕ	$\phi = p_2 \wedge a$	80°		
	г) кут між двома проектуючими променями	γ	$\gamma = p_1 \wedge p_2$	30°		
2	Віддаль між РЛС №1 та РЛС №2	a	задано	500 м		
3	Віддаль до "картинної" площини	H	задано	1000 м	1250 м	1500 м
4	Віддаль між базовою α та "картинною" β площинами по площині $\omega(p_1 \cap p_2)$ проектуючих променів	L	$L = \frac{H}{\sin \nu}$	1155 м	1330 м	1523 м
5	Віддаль від БПЛА до лінії a між РЛС №1 та РЛС №2	h	$h = L\left(1 - \frac{b}{a+b}\right)$	926 м		
6	Віддаль від БПЛА до РЛС №1	t_1	$t_1 = h_1\sqrt{1+(Ctg\delta)^2}$	985 м		
7	Віддаль від БПЛА до РЛС №2	t_2	$t_2 = h_1\sqrt{1+(Ctg\phi)^2}$	940 м		
8	Віддаль між проекціями проектуючих променів на "картинній" площині β	b	$b = \frac{L \cdot sin\gamma}{sin\delta \cdot sin\phi} - a$	124 м	218 м	322 м
9	Координати БПЛА: X	$X_к$	$X_к = t_1 \cos\delta$	337 м		
	Y	$Y_к$	$Y_к = h_1 \cdot \cos\nu$	463 м	317 м	162 м
	Z	$Z_к$	$Z_k = h_1 \cdot \sin\nu$	802 м	870 м	912 м

3.3. Розрахунок координат для визначення розташування БпЛА

Інший можливий варіант визначення місця розташування БПЛА у повітряному просторі – це розрахунок його координат $x_к$, $y_к$ та $z_к$, використовуючи запропоновані математичні залежності. Розглянемо приклад реалізації запропонованого способу. Нехай для спрощення радіолокаційні станції РЛС №1 та РЛС №2, а також точка С будуть

розташовані на однаковій географічній висоті. Тоді базова площина α(ABC), яку задають ці три точки A, B та C, буде уявною горизонтальною площиною. Нехай відстань a між радіолокаційними станціями становитиме a=|AB|=500м, а прогнозована висота польоту розшукуваного БПЛА знаходиться в межах від 800м до 950м. Уявно спроектуємо на розшукуваний БПЛА два проектуючі промені p_1 та p_2, кожен із яких починається у своїй точці облаштування радіолокаційної станції, тобто нехай РЛС №1≡$A \epsilon p_1$ та РЛС №2≡$B \epsilon p_1$. Приймемо різні значення кутів нахилу проектуючих променів до лінії a, що з'єднує між собою радіолокаційні станції, тобто нехай

$$\delta = \left| p_1 \wedge AB \right| = 70^\circ \text{ та } \phi = \left| p_2 \wedge AB \right| = 80^\circ .$$

Задамо віддаленість розшукуваного БПЛА від радіолокаційних станцій через кут нахилу ν утвореної проектуючими променями p_1 та p_2 площини $\omega(p_1 \cap p_2)$ до базової площини α(A,B,C) тобто $\nu = \omega \wedge \alpha$. Нехай ν=60˚. У розглянутому нами прикладі кути δ=70˚, ϕ=80˚ та ν=60˚ задаються, а в реальних ситуаціях числові значення цих кутів встановлюватимуться та фіксуватимуться на моніторах радіолокаційних станцій за напрямками векторів генерованих ними радіохвиль, що проходитимуть через розшукуваний БПЛА.

Тобто, умовно кажучи, в даному випадку неважливо, чи знаходиться розшукуваний безпілотник від радара, наприклад, кілометр, півтора або два. Натомість важливо, щоб пусковий промінь чітко показував напрямок розташування цього БПЛА, а наявні технічні засоби стабілізували його кути нахилу відносно прямої AB та базової площини α.

Задамо розташовану в повітряному просторі "картинну" площину β(W, S, L**)** на висоті, наприклад, H_1=1000 м, H_2=1250 м, H_3=1500 м.

Завдання полягає в тому, щоб розрахувати координати шуканого БПЛА в тривимірній декартовій системі координат, розташованої в початковій точці A≡РЛС№1. Розрахункові математичні залежності, вихідні дані та результати розрахунків наведені в таблиці. 1. Остаточні результати

розрахунків наведені в таблиці. 1 Координати розшукуваного безпілотника. Так для першого розглянутого варіанту при умовно заданому куті ν=60˚ нахилу утвореної проектуючими променями p_1 та p_2 площини $\omega(p_1 \cap p_2)$ до базової площини α(ΔABC) розраховані координати місцерозташування БПЛА становили:

$$K(X_к; Y_к; Z_к) = (337; 463; 802) \text{ м.}$$

Тобто висота його польоту становить $Z_к$=802м і він віддалений на t_1=985м від РЛС №1 та на t_2=940м від РЛС №2.

У другому варіанті при зміні кута до ν=70˚ координата $X_к$=337м залишилась незмінною, координата $Y_к$=317м зменшилася, а висота польоту зросла до $Z_к$=870м. Аналогічна закономірність спостерігається і при подальшому нарощуванні кута до ν=80˚ – координата $X_к$ незмінна, координата $Y_к$=162м зменшується, а висота польоту, відображена координатою $Z_к$=912м, наростає. Це свідчить про те, що одержані математичні залежності чутливо реагують на зміну вихідних умов та піддаються логічному трактуванню.

Оскільки математичні залежності, як слідує із наведених прикладів, доволі прості і виводяться із стереометрії та аналітичної геометрії, то створені для здійснення розрахунків програми не будуть складними. Проте вони повинні бути придатними до узгодження із програмним забезпеченням РЛС.

3.4. Висновки за розділом

В даному проаналізовано метод визначення положення БПЛА в повітряному просторі за допомогою кінематичного плану, який базується на розрахунку відстані літака від радіолокаційних станцій у декартовій системі координат, нечутливий до хибних зовнішніх сигналів, відбитих від літака. З цієї причини рельєф місцевості є більш точним порівняно з методом різниці дальностей та іншими методами радіопошуку місцезнаходження літака.

Крім визначення координат, швидкостей і напрямків космічних переміщень БПЛА, метод кінематичного проектування придатний для визначення місцезнаходження рухомих об'єктів на поверхні суші та води, а також на глибокій воді.

4 РОЗРОБКА ДИНАМІЧНОГО СЕРЕДОВИЩА ДЛЯ ПРОГРАМИ МОДЕЛЮВАННЯ

4.1. Вибір середовища моделювання

Симуляція польоту літака буде виконана в зоні KBOI для зручності використання в середовищі розробки Unity. Unity також має вбудований 2D-шаблон, який полегшує імітацію роботи радіолокаційної станції. А програму напишемо мовою C# середовища програмування Visual Studio.

Unity [6] — кросплатформне середовище розробки ігор, розроблене американською компанією Unity Technologies. Unity дозволяє створювати програми, які працюють на більш ніж 25 різних платформах, включаючи ПК, ігрові консолі, мобільні пристрої, веб-додатки тощо. Unity була випущена в 2005 році і з тих пір постійно розвивається.

Як правило, ігровий движок пропонує багато функцій, які дозволяють використовувати його в різних іграх, зокрема імітацію фізичного середовища, нормальні карти, динамічні тіні тощо. На відміну від багатьох ігрових движків, Unity має дві основні переваги: наявність візуального середовища розробки та підтримку різних платформ. Перший фактор включає не лише інструментарій візуального моделювання, а й інтегроване середовище, ланцюжок складання з метою підвищення продуктивності розробників, особливо на етапах прототипування та тестування. Кросплатформна підтримка забезпечує не тільки місця розгортання (встановлення на ПК, мобільний пристрій, консоль тощо), але й доступність засобів розробки (інтегроване середовище можна використовувати як під операційними системами Windows, так і під Mac OS).

Серед недоліків названі обмеження візуального редактора при роботі з багатокомпонентними конструкціями, оскільки візуальна робота ускладнюється у складних сценах. Другий недолік – відсутність опори Unity посилається на зовнішні бібліотеки, які програмістам доводиться

налаштовувати самостійно, що також ускладнює командну роботу. Інший недолік пов'язаний з використанням зразків (заздалегідь виготовлених) шаблонів. З одного боку, ця концепція Unity пропонує гнучкий підхід до візуального редагування об’єктів, але з іншого боку, редагувати такі шаблони складно. Крім того, версія двигуна WebGL має низку невирішених проблем із продуктивністю, споживанням пам’яті та продуктивністю мобільних пристроїв через її архітектурні особливості (переклад коду з C# на C++, а потім JavaScript).

Проект в Unity поділений на сцени (рівні) [7] - окремі файли, що містять власні ігрові світи зі своїм набором об'єктів, сценаріїв і налаштувань. Сцени можуть містити як реальні об’єкти (моделі), так і порожні ігрові об’єкти – об’єкти, які мають моделі («останні»). Об’єкти, у свою чергу, містять набір компонентів, з якими взаємодіють скрипти. Об’єкти також мають ім’я (в Unity дозволено два або більше об’єктів з однаковими іменами), можуть мати мітку та шар для відображення. Тому кожен об’єкт у сцені повинен мати компонент Transform – він зберігає координати положення, обертання та розміри об’єкта вздовж усіх трьох осей. Об’єкти з видимою геометрією також мають за замовчуванням Mesh Renderer, який робить модель об’єкта видимою.

Поведінка ігрових об'єктів управляється за допомогою підключених до них компонентів. Хоча вбудовані компоненти Unity можуть бути дуже універсальними, ви незабаром побачите, що вам потрібно вийти за межі їхніх можливостей, щоб реалізувати власні унікальні функції ігрового процесу. Unity дозволяє створювати власні компоненти за допомогою скриптів (рис. 4.1.). Вони дозволяють запускати ігрові події, змінювати налаштування компонентів і реагувати на введення користувача будь-яким способом.

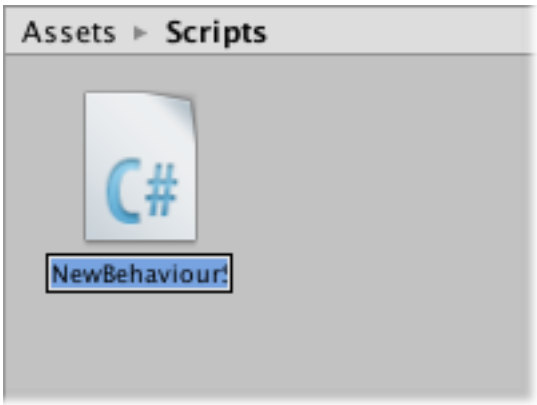


Рисунок 4.1 - Новий скрипт на мові C#.

Цей сценарій взаємодіє з внутрішніми механізмами Unity, створюючи клас, який успадковує внутрішній клас під назвою MonoBehaviour. Ви можете розглядати клас як своєрідний проект для створення нового типу компонента, який можна приєднати до об’єкта гри. Щоразу, коли ви додаєте компонент сценарію до ігрового об’єкта, створюється новий екземпляр об’єкта, визначеного планом. Ім'я класу береться з імені, яке ви вказали під час створення файлу. Ім’я класу та ім’я файлу мають збігатися, щоб підключити компонент сценарію до об’єкта гри.

Найважливішими речами, на які слід звернути увагу, є дві функції, визначені в класі. Функція оновлення — це місце для розміщення коду, який виконує оновлення фрейму для об’єкта гри. Це може бути рух, тригерні дії та реакції на введення користувача, загалом усе, що потрібно обробити з часом у ігровому процесі. Щоб дозволити функції оновлення виконувати свою роботу, часто корисно ініціалізувати змінні, перерахувати властивості та спілкуватися з іншими об’єктами гри, перш ніж виконувати будь-які дії. Функція Start викликається Unity перед початком ігрового процесу (тобто перед першим викликом функції Update) і є найкращим місцем для ініціалізації. Unity підтримує мову програмування C# з самого початку. C# (вимовляється C#) — стандартна мова, подібна до Java або C++. C# — це те, що ми будемо використовувати для створення сценарію для нашого радара.

C# відноситься до сімейства мов, які мають C-подібний синтаксис, їх синтаксис близький до C++ і Java. Мова статично типізована, починаючи від

поліморфізму, завантаження операторів (включаючи оператори явного та неявного приведення типу), делегатів, властивостей, подій, змінних, атрибутів, типів і узагальнених методів, ітераторів, анонімних функцій із підтримкою пакетів, LINQ, винятків, коментарів у Формат XML.

XML — це розширена мова розмітки. Мова називається розширюваною, тому що вона не змінює розмітку, яка використовується в документах: розробник може вільно створювати розмітку відповідно до потреб конкретного регіону та обмежений лише синтаксичними правилами мови.

Завдяки значній адаптації своїх попередників - C++, Delphi, Module, Smalltalk і особливо Java - мови C#, спираючись на практику їх використання, деякі моделі, які виявилися проблематичними при розробці програмних систем, видаляють Наприклад, C#, на відміну від C++, не підтримує багаторазове успадкування класів (дозволяючи при цьому декілька реалізацій інтерфейсів).

C# було розроблено як мову програмування на прикладному рівні для CLR (Common Language Runtime Environment) і як така залежить насамперед від можливостей самого CLR. Це стосується насамперед системи типів C#, яка відображає BCL (Standard Class Library). Наявність або відсутність певних експресивних мовних особливостей залежить від того, чи можна перекласти конкретну мовну функцію у відповідні конструкції CLR. Таким чином, оскільки CLR еволюціонував від версії 1.1 до 2.0, сам C# був значно збагачений. Таку взаємодію слід очікувати в майбутньому (проте цю схему було порушено з випуском C# 3.0, яке є розширенням мови, яке не покладається на розширення платформи .NET). CLR надає C#, як і інші .net-орієнтовані мови мають багато можливостей, яких не вистачає «класичним» мовам програмування. Наприклад, збирання сміття не реалізовано в самій C#, але обробляється CLR для програм, написаних на C#, так само, як і для програм VB.NET, J# тощо.

В якості середовища програмування для мови C# буде використовуватися Visual Studio.

Visual Studio — це платформа для написання, налагодження та компіляції коду, а також для публікації ваших програм. На додаток до стандартного редактора та відладчика, які є в більшості IDE (інтегрованих середовищ розробки), Visual Studio містить компілятори, інструменти завершення коду, графічні дизайнери та багато інших функцій для покращення процесу розробки.

Visual Studio [7] містить редактор вихідного коду з підтримкою технології IntelliSense і можливістю найпростішої регенерації коду. Вбудований налагоджувач може працювати як налагоджувач на рівні джерела, так і на рівні машини. Інші вбудовані інструменти включають редактор форм для спрощення створення GUI програми, веб-редактор, конструктор класів і конструктор схем бази даних. Visual Studio дозволяє додавати сторонні надбудови (плагіни) для розширення функціональності майже на будь-якому рівні, включаючи додавання підтримки систем контролю версій вихідного коду (таких як Subversion і Visual SourceSafe), додавання нових наборів інструментів (наприклад, для редагування Visual і дизайн (код на об'єктно-орієнтованих мовах програмування) або інструменти для інших аспектів процесу розробки програмного забезпечення (наприклад, клієнт Team Explorer для роботи з Team Foundation Server).

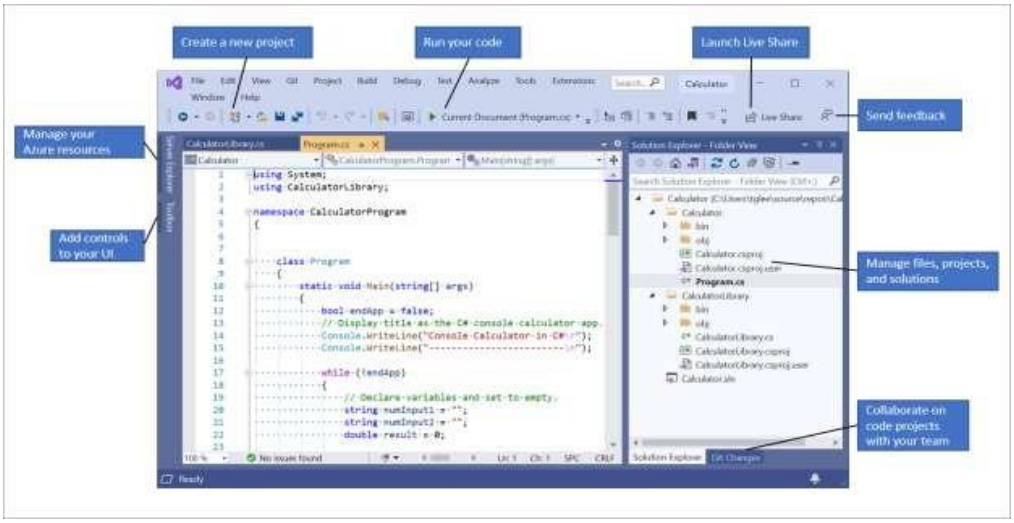


Рисунок 4.2 – Visual Studio з відкритим проектом.

На рис. 4.2. Представляє середовище Visual Studio з відкритим проектом і порадами щодо основних вікон і функцій.

У верхньому правому куті вікна переглядача рішень можна переглядати файли коду, навігацію та керувати ними. Переглядач рішень дозволяє вам упорядковувати свій код, групуючи файли в рішення та проекти.

Вміст файлу відображається в центральному вікні найбільш використовуваного редактора. У вікні редактора можна вносити зміни в код або створювати інтерфейс користувача, наприклад вікно з кнопками або текстовими полями.

Вікно Git Changes у нижньому правому куті дозволяє відстежувати робочі елементи та ділитися кодом за допомогою Git, GitHub або інших технологій керування версіями.

4.2. Моделювання функціональності індикатора колового огляду

Ми створюємо новий 2D проект. Перш ніж симулювати польоти дронів, нам спочатку потрібно створити кругле вікно перегляду та інші об’єкти на сцені, такі як сценарії та колекції, які містять зображення наших майбутніх дронів.

Спочатку нам потрібно створити площину для використання в якості фону. На ньому ми відображаємо зображення індикатора кругового огляду, яке знаходимо в Інтернеті [16]. Для цього натисніть праву кнопку миші та виберіть у вікні тривимірний об’єкт площини. Відредагуйте розмір і положення площини так, щоб вона була перпендикулярна камері. Після цього ми копіюємо зображення маркера в папку ресурсів проекту, тепер ми можемо прикріпити зображення маркера до площини, що ми робимо це. Результат показано на рис. 4.3.

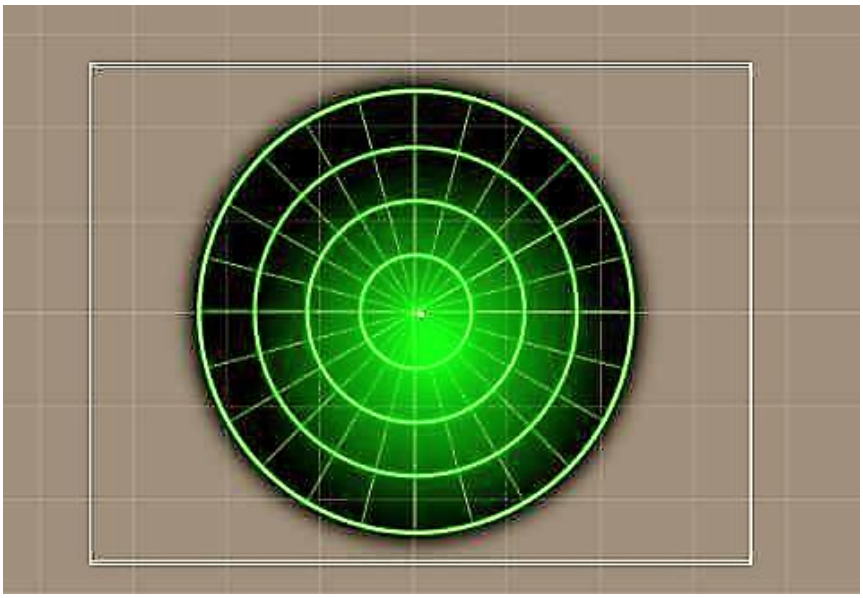


Рисунок 4.3 - Зображення індикатора кругового огляду.

Наступний об’єкт, який ми створимо, — це порожній об’єкт, який ми розмістимо в центрі нашого маркера, це буде точка, яку шукають наші дрони, з деяким розширенням, яке ми розмістимо за ним пізніше.

За допомогою сценаріїв нам це потрібно, щоб деякі з наших об’єктів, наприклад дрони типу літака, не поверталися на 180 градусів і автоматично віддалялися. І все ж вони випадково перетнули шлях радіолокаційної станції.

Тепер давайте створимо ще один порожній об'єкт, це буде наш дрон ресурс. Ми створюємо для нього скрипт на C# і прив’язуємо його до порожнього об’єкта. Ми вставляємо параметри надсилання дрона, як і інші параметри, у код скрипта. Давайте створимо ще один подібний об'єкт зі скриптом для другого типу дрона - вертольота. Розбиваємо їх на різні об’єкти, щоб у подальшому контролювати кількість дронів кожного типу окремо, а також інші параметри. Усі ці об’єкти ми створюємо у вікні «Ієрархія» (рис. 4.4).

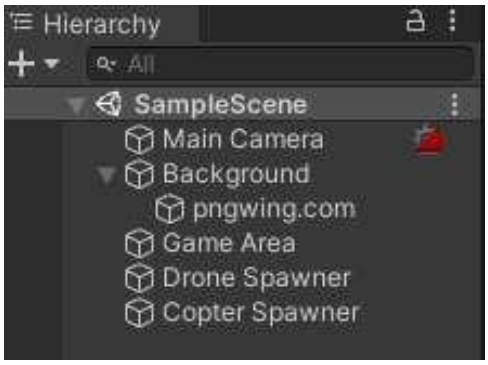


Рисунок 4.4 – Вікно «Ієрархія»

Нарешті, нам потрібні ще два порожні об’єкти, щоб прикріпити до них зображення дрона. Точніше, як це виглядає в нашому індикаторі кругового огляду. У нашому випадку це буде червона точка, щоб не зливатися із зеленим фоном нашого радара. Ми також створюємо скрипт для цих об’єктів, для кожного з дронів будуть призначені завдання, коли вони з’являться в додатку. Ми створюємо два об’єкти, тому що дрони матимуть два різні сценарії: один для типу літака та інший для типу гелікоптера.

Ми починаємо з копіювання зображення червоної крапки в нашу вихідну папку, а потім прикріплюємо його до обох порожніх об’єктів. Тепер ми бачимо, що крапки з’явилися на головному екрані, щоб вони зникли, нам потрібно перемістити порожні об’єкти до папки ресурсів, щоб вони це не заважатиме функціональності програми, і в той же час вона працює як слід. Тепер у нашій папці з ресурсами буде 2 об’єкти для відображення дронів, 4 скрипти (2 для створення дронів і 2 для переміщення дронів), а також 2 зображення маркера та червоної точки (рис. 4.5).

Рисунок 4.5 - Папка з ресурсами

Коли побудова проекту закінчена і файли з’єднані, час писати код, для початку переходимо в скрипт SCR_DroneSpawner, де створюємо нові дрони

та їх параметри.

4.3. Висновки за розділом

Описано існуюче програмне забезпечення для керування БПЛА, а також наведено архітектуру проекту та опис його компонентів.

Під час створення динамічного середовища середовище розробки Unity надало можливість зосередитися саме на логіці роботи модуля, що дозволило зробити розробку швидшою та приємнішою. Також середовище програмування Visual Studio та мова прграмування C# повністю оправдано називається «простии та швидкими», так як на освоєння роботи з ними пішло відносно мало часу, а наявність великого рівня підтримки середовища розробки Unity розробниками та велика кількість інформації про роботу з нею залишили приємний досвід розробки.

5 МОДЕЛЮВАННЯ РУХУ БПЛА В ЗАДАНОМУ СЕРЕДОВИЩІ

На основі згаданого файлу траєкторії керується графічне зображення БПЛА під час демонстрації. Щоб дрон в потрібний момент почав рухатися разом із трекером, на одну з кнопок клавіатури покладено запуск автоматичного керування. Сам факт натискання кнопки та оновлення деяких початкових значень обробляється у функції handleInput() демонстраційного стану, а решта логіки польоту обробляється у функції стану update().

Автоматичне переміщення забезпечується логічною зміною mv, яка при значенні true дозволяє дрону працювати автоматично, а при значенні false функція, що відповідає за автоматичний рух логіки польоту БПЛА, не активна. . Якщо вказано, що швидкість автоматичного польоту має бути збільшена (mv==true), то перше, що модуль викликає, це копія протоколу через функцію floatFromVector(), вказана як один із аргументів як необхідні дані.

Це працює наступним чином: щоб знайти потрібне значення через цю функцію, знадобиться знання структури протоколу. Як уже згадувалося, дані трекера у файлі розділені на три стовпці, а саме момент (секунди) з моменту початку польоту, координати x (метри) та y (метри) положення дрона в цей момент. Щоб пройти через такий набір даних, вам потрібно знати, перш за все, положення дрона, і це еквівалентно номеру рядка в протоколі, оскільки кожен список для свого моменту часу (точка часу, x, y) Рядок із відступами — це регістр. І дані переміщуються через такий «набір» за допомогою номера позиції (цілочисельний параметр targ_pos).

Але якщо ви знаєте номер потрібної лінії (потрібний стан дрона), вам все одно потрібно вибрати один з трьох параметрів: момент часу, x або y цього стану. Для цього стане в нагоді параметр функції floatFromVector(), тобто ціле число targ_col, тобто номер стовпця. І так як структура протоколу визначена, стовпець номер 1 відповідатиме за момент часу, номер 2 – за

координату x, а номер 3 – за координату y. Таким чином можна отримати необхідні дані з протоколу за номером позиції та номером графи.

Також зауважте, що копія протоколу, з яким працює функція floatFromVector(), є набором символів. А символьний тип даних не підходить для подальших обчислень, тому перед поверненням результату символьний тип перетворюється в дробове число потрібного типу у функції floatFromVector(). Слід також зазначити, що для того, щоб створити бажане значення, перетворення набору символів у рядок символів спочатку виконується шляхом поступової конкатенації (злиття рядків). Коли значення форматується як рядок символів, воно вже перетворюється на дробове число стандартною функцією stuff().

Тепер, коли floatFromVector() надає моделі згенеровані дані, потім розраховується для переміщення БПЛА. Функція moveToPoint() відповідає за переміщення графічного об'єкта БПЛА по вузлових точках (тобто по тих, координати яких передбачені в протоколі). Крім різноманітних лічильників і логічних змінних, головне, що приймає функція, це час переміщення від першої точки до другої, поточні координати дрона та координати наступної точки для переміщення. Отже, якщо початкову і кінцеву координати для двох вузлових точок можна відразу зчитати з протоколу, то для визначення часу прольоту між цими точками необхідно відняти моменти часу перебування в точці A (tA). Момент прибуття в пункт B (tB). Отже, різниця моментів у між двома вузловими точками (t) представлятиме час, необхідний для польоту.

Важливо відзначити, що автоматичний політ БПЛА відбувається поетапно, тобто функція moveToPoint() викликається циклічно в момент досягнення дроном вузлових точок і працює за принципом конвеєра. Точка B попередньої частини шляху стає точкою A наступної, за винятком першої та останньої координат усієї траєкторії польоту. Але якщо дрон рухався тільки уздовж вузлових точок, без можливості з'єднання між ними, і не з'являвся на них відразу, то плавність польоту і всієї демонстрації залежить від «щільності» значення наданих даних.

Модель розроблена таким чином, що навіть якщо відстань між точками настільки велика, що протокол користувача побачить поштовхи під час руху БПЛА у вигляді даних низької «щільності», такі дані будуть повними. програмне забезпечення, і дрон буде легко переміщатися з точки в точку. Як це досягається?

В основному розраховується середнє значення координат точок, куди повинен рухатися дрон. Функція interpolate() відповідає за роботу з векторами руху через інструментарій бібліотеки SFML. Таким чином, в даному випадку ширина руху в двовимірному просторі має два значення: вертикальну і горизонтальну координати. Для виконання роботи створюються два вектори руху: вектор A (містить координати першої вузлової точки) і вектор B (містить координати другої вузлової точки). Далі вектор A віднімається від вектора B і їх різниця множиться на лічильник факторів, який змінює координати положення графічного об’єкта з кожним циклом. Отриманий вектор потім додається до вектора A, щоб розраховані координати були точно пов’язані з початковою точкою. У функції moveToPoint() interpolate() надає координати, у яких має з’явитися дрон, за допомогою методу SFML setPosition(), який використовує бібліотеку для позиціонування спрайту дрона у вказаних координатах. Крім того, слід розуміти, що траєкторія польоту дрона — це набір точок, як вузлових, які визначені в протоколі, так і проміжних точок, модель керування рухом яких розраховує сама.

Тоді, коли дрон досягає кінцевої точки сегмента, функція moveToPoint() отримує нові дані для нової кінцевої точки руху вздовж сегмента шляху, а початкова точка є кінцевою точкою попереднього сегмента, а лічильник interpolate() Функція скидається до нуля, щоб обчислити новий набір точок на наступному сегменті шляху. Це робиться до тих пір, поки в протоколі не буде відсутня наступна точка, після чого БПЛА перестає рухатися зі швидкістю.

5.1. Код створення дронів

Спочатку ми пишемо бібліотеки, які будемо використовувати в нашому коді. У нашому випадку нам не потрібно нічого додавати, дженерики (System.Collections) і основна бібліотека Unity (UnityEngine) були описані автоматично. Нижче наведено фрагмент коду, який описує бібліотеки.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
```

Тепер давайте створимо компоненти, в які внесемо інформацію про кількість дронів, швидкість, висоту і т.д. Ми напишемо весь наступний код у класі SCR_DroneSpawner. Нижче наведено фрагмент коду, який створює компоненти для класу SCR_DroneSpawner.

```
public class SCR_DroneSpawner : MonoBehaviour
{
public GameObject game_area;
public GameObject drone_prefab;
public int drone_count1 = 0;
public int drone_limit1 = 5;
public int drone_per_frame = 1;
public float spawn_circle_radius = 25.0f;
public float death_circle_radius = 26.0f;
public float fastest_speed = 9.8f;
public float slowest_speed = 1.5f;
public float max_altitude = 500.0f;
public float min_altitude = 5.0f;
```

У першому рядку ми пишемо game_area в нашому сценарії, щоб отримати інформацію про місцезнаходження нашого об’єкта. У наступному рядку ми створюємо шаблон, з якого беремо зображення дронів. drone_count1 відповідає за поточну кількість дронів, цей компонент не можна редагувати під час роботи програми. drone_limit1 відповідає за максимальну кількість дронів, це компонент, який буде використовуватися

для зміни кількості дронів. drone_per_frame відповідає за те, скільки дронів генерується на кадр.

Щоб наші дрони з’являлися навколо країв круглого вікна перегляду, нам потрібно створити коло, краї якого ми використовуємо як точки появи дронів. Для цього нам потрібні два компоненти spawn_circle_radius і death_circle_radius. У них записані радіуси кіл, одне коло використовується для створення дронів, а інше для видалення дронів, які перевищують маркер. Щоб уникнути автоматичного знищення дронів під час будівництва, ми трохи збільшуємо радіус другого кола.

Ми редагуємо швидкість дрона за допомогою двох інших компонентів. У компоненті fastest_speed ми записуємо максимальну швидкість, а в компоненті slowest_speed — мінімальну швидкість. Надалі між цими двома числами програма буде випадковим чином вибирати швидкість кожного дрона окремо. Згідно з класифікацією дронів, ми знаємо, що максимальна швидкість польоту може становити 980 км/год, припустимо 9,8 на максимальній швидкості. Мінімальна швидкість для дронів авіаційного типу становить 150-400 км/год, для середньо швидкісного дрона ми встановлюємо значення slowest_speed на 1,5.

Останні два компоненти – це мінімальна та максимальна висота дрона (max_altitude та min_altitude). Програма також випадковим чином вибирає висоту кожного дрона. Середній діапазон робочих висот авіаційних безпілотників становить 50-5000 метрів, тому ми закладаємо 5-500 у відповідні компоненти. Рис. 5.1. Він показує створені нами компоненти та їхні значення.

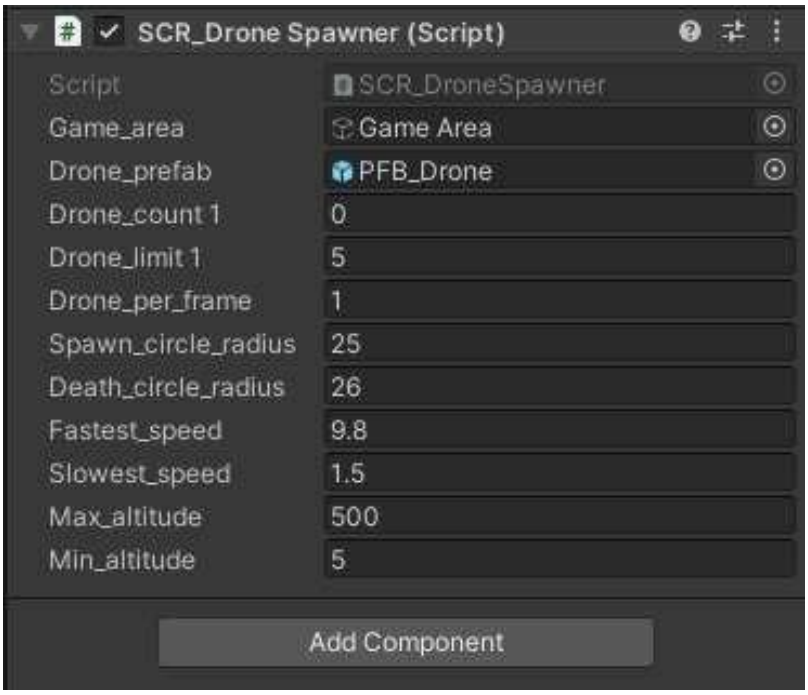


Рисунок 5.1 - Компоненти сценарію SCR_DroneSpawner

Нижче наведено фрагмент коду з функціями Start() і Update()

```
void Start()
{
}
void Update()
{
MaintainPopulation();
}
```

Наша функція Start() буде порожньою. І ми будемо писати у функції Update().

Майбутня функція MaintainPopulation(), за допомогою якої ми будемо створювати нові дрони без збільшення встановленого раніше максимального ліміту. Нижче наведено фрагмент коду з функцією MaintainPopulation().

```
void MaintainPopulation()
{
if (drone_count1 < drone_limit1)
{
for (int i = 0; i < drone_per_frame; i++)
{
Vector3 position = GetRandomPosition();
SCR_Drone drone_script = AddDrone(position);
```

```
    drone_script.transform.Rotate(Vector3.forward   *   Random.Range(-
45.0f,45.0f));
    }
    }
    }
```

У функції MaintainPopulation() ми контролюємо кількість дронів у зоні, якщо кількість дронів менше drone_limit1, ця функція спочатку знаходить випадкову позицію в області, створює дрон, а потім повертає його з середини. З області, яка повертається під довільним кутом до 45 градусів ліворуч або праворуч. за допомогою Random.Range(). це необхідно тільки для дронів авіаційного типу, щоб вони літали в зоні маркера, а не на трасі. Нижче наведено фрагмент коду з функцією GetRandomPosition().

```
Vector3 GetRandomPosition()
{
Vector3 position = Random.insideUnitCircle.normalized;
position *= spawn_circle_radius;
position += game_area.transform.position;
return position;
}
```

Функція GetRandomPosition вибирає випадкову точку для нерестового дрона за допомогою двовимірного кола по краях області маркера. Це стане можливим за допомогою Random.insideUnitCircle.normalized. Якщо ми просто напишемо Random.insideUnitCircle, то наші дрони з’являться не тільки по краях, але і в середині кола. Але якщо додати нормалізовані, то трутні з’являться лише на краях кола (рис. 5.2).

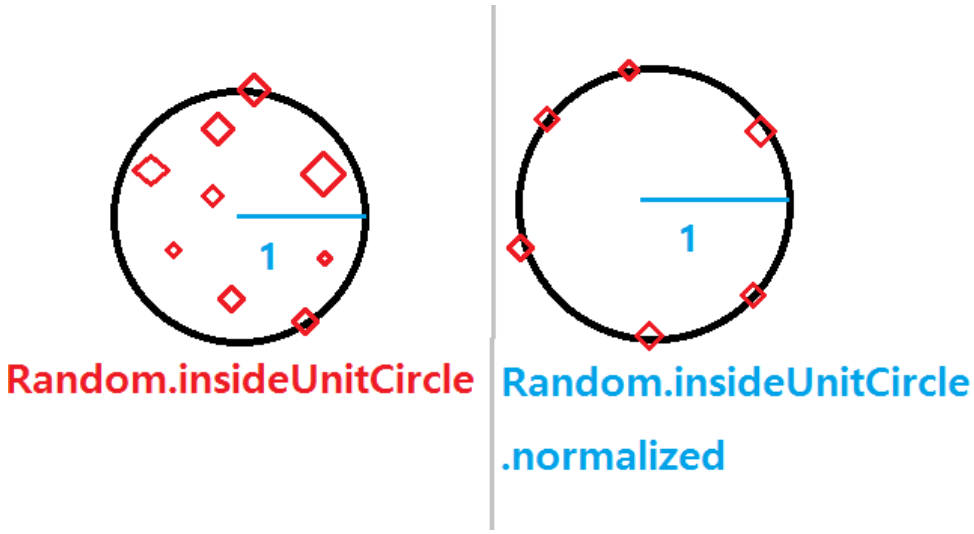


Рисунок 5.2 - Спавн з використанням normalized

Нижче наведено фрагмент коду з функцією AddDrone().

```
SCR_Drone AddDrone(Vector3 position)
{
drone_count1 += 1;
GameObject new_drone = Instantiate(
drone_prefab,
position,
Quaternion.FromToRotation(Vector3.up,
(game_area.transform.position - position)),
gameObject.transform
);
SCR_Drone drone_script = new_drone.GetComponent<SCR_Drone>();
drone_script.drone_spawner = this;
drone_script.game_area = game_area;
drone_script.speed = Random.Range(slowest_speed, fastest_speed);
drone_script.altitude = Random.Range(min_altitude, max_altitude);
return drone_script;
}
}
```

Останньою функцією цього скрипта є створення дронів типу літака буде AddDrone. Тут буде код для створення дронів і налаштування основних параметрів цих дронів. Спочатку ми додаємо +1 до кількості дронів, щоб дронів не було забагато. Потім ми прикріплюємо зображення з вихідної папки, позицію в регіоні до нового дрона і також обертаємо його, щоб він став

центром регіону.

Після створення дрона ми можемо встановити основні параметри. Використовуючи Random.Range, ми потім встановлюємо швидкість і висоту нашого дрона повертаємо всі ці значення в SCR_Drone, щоб інший скрипт міг отримати доступ до цих даних. Ми можемо переходити до сценарію породження вертолітних дронів.

5.2. Код спавнера коптерів

Нижче наведено фрагмент коду, який описує бібліотеки та створення компонентів класу SCR_CopterSpawner.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class SCR_CopterSpawner : MonoBehaviour
{
public GameObject game_area2;
public GameObject drone_prefab;
public int copter_count = 0;
public int copter_limit = 10;
public int drone_per_frame = 1;
public float spawn_circle_radius = 25.0f;
public float death_circle_radius = 26.0f;
public float fastest_speed = 2.0f;
public float slowest_speed = 0.5f;
public float max_altitude = 100.0f;
public float min_altitude = 1.0f;
```

Ми налаштовуємо ті самі компоненти та змінюємо лише номери на них, які є більш доцільними для гелікоптерних дронів (зниження максимальної швидкості). Для встановлення мінімальної та максимальної швидкості ми знову використовуємо класифікацію дронів, на даний момент беремо швидкість 50 км від малошвидкісних дронів, максимальна швидкість польоту яких не перевищує 200 км/год. /год як мінімум.

Висота польоту буде меншою, ніж у безпілотника літакового типу. Середній робочий діапазон висот вертолітних дронів становить 10-1000 м, ми змінюємо значення висоти на основі цих даних (рис. 5.3). Він показує створені нами компоненти та їхні значення.

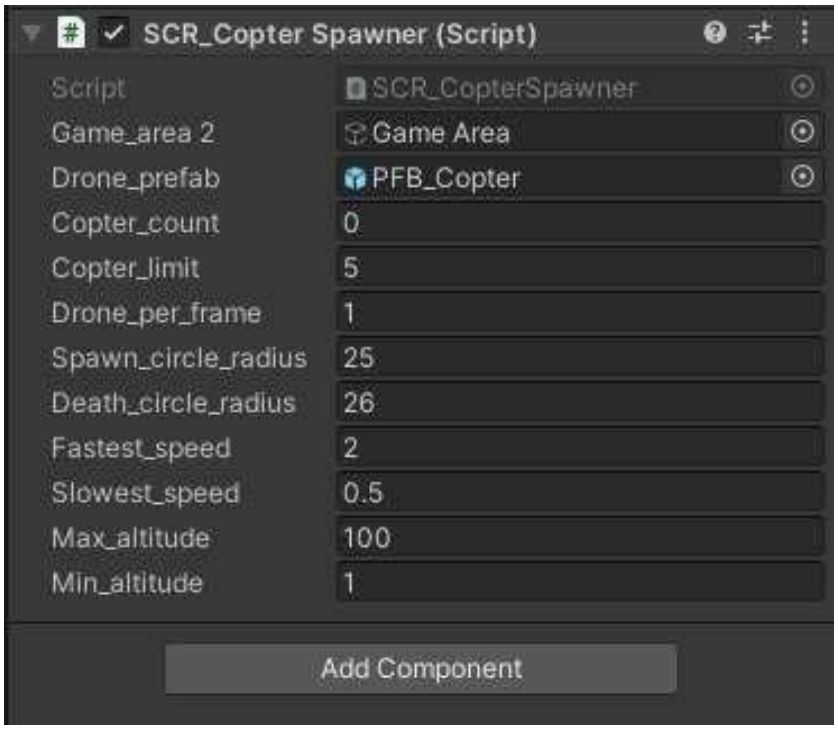


Рисунок 5.3 - Компоненти сценарію SCR_CopterSpawner

Нижче наведено частину коду з Start(), Update() і функціями MaintainPopulation2().

```
void Start()
{
}
void Update()
{
MaintainPopulation2();
}
void MaintainPopulation2()
{
if (copter_count < copter_limit)
{
for (int i = 0; i < drone_per_frame; i++)
{
Vector3 position = GetRandomPosition();
SCR_Drone2 drone_script2 = AddCopter(position);
```

```
    }
    }
    }
```

Також прибираємо поворот дрона на 45 градусів, тому що міняти нічого

На цей раз гелікоптер літатиме за допомогою скрипта SCR_Drone2. Ось тут ми змінимо курс. Нижче наведено фрагмент коду з функціями GetRandomPosition() і AddCopter().

```
    Vector3 GetRandomPosition()
    {
    Vector3 position = Random.insideUnitCircle. normalized;
    position *= spawn_circle_radius;
    position += game_area2.transform.position;
    return position;
    }
    SCR_Drone2 AddCopter(Vector3 position)
    {
    copter_count += 1;
    GameObject new_drone = Instantiate(
    drone_prefab,
    position,
    Quaternion.FromToRotation(Vector3.up,
(game_area2.transform.position -
    position)),
    gameObject.transform
    );
    SCR_Drone2 drone_script2 = new_drone.GetComponent<SCR_Drone2>();
    drone_script2.copter_spawner = this;
    drone_script2.game_area2 = game_area2;
    drone_script2.speed = Random.Range(slowest_speed, fastest_speed);
    drone_script2.altitude        =        Random.Range(min_altitude,
max_altitude);
    return drone_script2;
    }
    }
```

Решта коду залишається такою, якою є в SCR_DroneSpawner Створюємо випадкову позицію в колі. Ми збираємо наш дрон і налаштовуємо його параметри, зовнішній вигляд, кут огляду, швидкість і висоту.

Коли написання коду для налаштування параметрів і створення БПЛА закінчено, необхідно змоделювати політ цих БПЛА. Почнемо з коду типу дрона.

5.3. Код літакових дронів

Нижче наведено фрагмент коду, який описує створення бібліотек і компонентів класу SCR_Drone.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class SCR_Drone : MonoBehaviour
{
public SCR_DroneSpawner drone_spawner;
public GameObject game_area;
public float speed;
public float altitude;
```

Перші два рядки в нашому класі з’єднують цей сценарій зі сценарієм SCR_DroneSpawner і з маркером game_area. Потім ми робимо значення швидкості та висоти загальнодоступними, щоб ми могли бачити їх у скрипті значка. На рис. 5.4. Показані значення, створені програмою для одного з дронів.

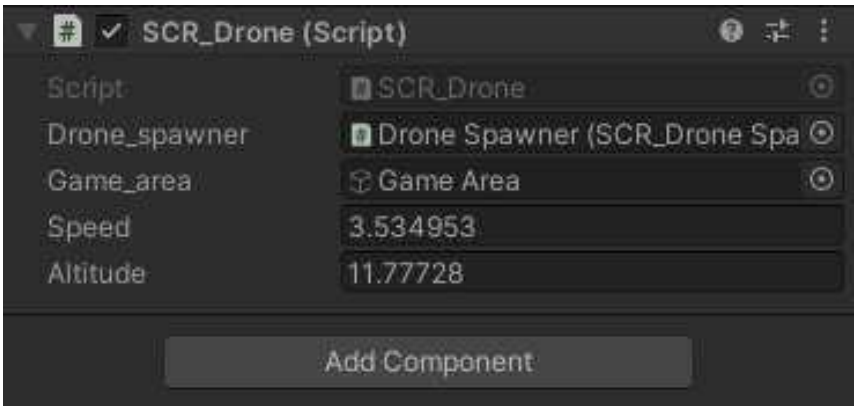


Рисунок 5.4 - Швидкість і висота одного з дронів

Нижче наведено фрагмент коду з функціями Update() і Move(). void

Update()

```
{
Move();
}
void Move()
{
transform.position += transform.up * (Time.deltaTime * speed);
float distance = Vector3.Distance(transform.position,
game_area.transform.position);
if (distance > drone_spawner.death_circle_radius)
{
RemoveDrone();
}
}
```

Заради швидкості ми встановили напрямок його польоту в SCR_DroneSpawne, тому все, що нам потрібно зробити, це написати код, щоб видалити дрон, якщо він летить за межі нашого кола смерті. Для цього ми створимо ще одну функцію RemoveDrone(). Нижче наведено фрагмент коду з функцією RemoveDrone().

```
void RemoveDrone()
{
Destroy(gameObject);
drone_spawner.drone_count1 -= 1;
}
}
```

Ця функція зменшує значення drone_count1 на одиницю та знищує об’єкт. Переходимо до дронів-вертольотів.

5.4. Код коптерів

Нижче наведено фрагмент коду, який описує бібліотеки та створення компонентів класу SCR_Drone2.

```
using System.Collections;
```

```
using System.Collections.Generic;
using UnityEngine;
public class SCR_Drone2 : MonoBehaviour
{
public float moveDuration = 4f;
public float pauseDuration = 2f;
public float timer;
public bool paused;
private Vector3 movementDirection;
private Vector3 movementPerSecond;
public SCR_CopterSpawner copter_spawner;
public GameObject game_area2;
public float speed;
public float altitude;
```

Основна відмінність дронів авіаційного типу від вертольотів полягає в здатності цього дрона зупинятися в польоті і змінювати напрямок. Щоб симулювати політ таких дронів, нам потрібен таймер, який змінює наш об’єкт з руху на зупинку і навпаки.

Спочатку давайте напишемо два компоненти moveDuration, які відповідають за те, скільки секунд рухається дрон. А також PauseDuration, який зберігає час, протягом якого дрон залишається нерухомим. Ми створюємо сам таймер і публікуємо його, щоб побачити, скільки часу залишилося. Стверджує, що наш умовний вираз призупиниться. MoveyDirection відповідає за зміну напрямку дрона, а PerseCond — за швидкість. Решту залишаємо без змін. На рис. 5.5. Показані значення, створені програмою для одного з дронів.

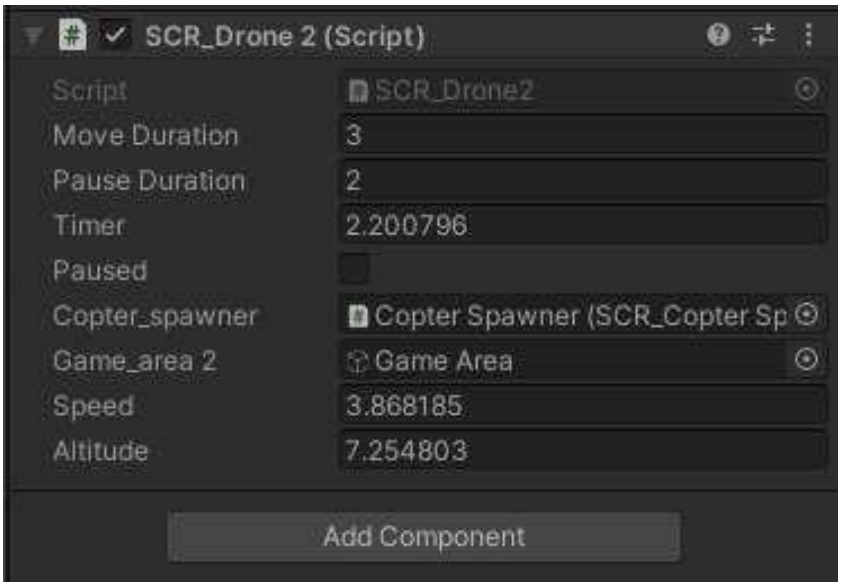


Рисунок 5.5 – Усі компоненти гелікоптерного дрона

Нижче наведено фрагмент коду з функцією Start().

```
void Start()
{
timer = moveDuration; //to get it moving from the start
paused = false;
FindNewDirection();
}
```

Щоб дрон входив із самого початку програми функція Start() вмикає таймер, і ми викликаємо функцію FindNewDirection(), щоб вибрати напрямок і рухатися. Нижче наведено фрагмент коду з функцією Update().

```
void Update()
{
timer -= Time.deltaTime;
if (timer <= 0)
{
if (paused)
{
timer = moveDuration;
FindNewDirection();
paused = false;
}
else
{
```

```
timer = pauseDuration;
paused = true;
}
return;
}
if (!paused)
{
transform.position = new Vector3(transform.position.x +
(movementPerSecond.x * Time.deltaTime),
transform.position.y + (movementPerSecond.y * Time.deltaTime));
}
Move2();
}
```

У функції Update() ми пишемо наш код таймера так, щоб це працювало протягом усієї програми. Якщо паузи немає, таймер руху веде зворотний відлік і переміщує дрон. Коли пауза ввімкнена, таймер зменшить паузу та вимкне паузу в кінці. Функція Move2() знищує дрон, лише якщо він вилітає за межі кола смерті. Нижче наведено фрагмент коду з функціями Move2(), Delete() і FindNewDirection().

```
void Move2()
{
float distance = Vector3.Distance(transform.position,
game_area2.transform.position);
if (distance > copter_spawner.death_circle_radius)
{
RemoveDrone();
}
}
void RemoveDrone()
{
Destroy(gameObject);
copter_spawner.copter_count -= 1;
}
void FindNewDirection()
{
movementDirection = new Vector3(Random.Range(-1.0f, 1.0f),
Random.Range(-1.0f, 1.0f)).normalized;
```

```
movementPerSecond = movementDirection * speed;
}
}
```

FindNewDirection() генерує випадковий вектор напрямку з величиною 1, потім регулює швидкість нашого дрона.

Зберігаємо готовий код і потім перевіряємо працездатність програми. Давайте натиснемо кнопку «Грати», почнеться ця симуляція. Якщо потрібно, натисніть кнопку «Пауза» праворуч від кнопки «Відтворити», щоб зупинити програму. Повторне натискання «Відтворити» призведе до зупинки симуляції (Рис. 5.6).

Рисунок 5.6 – Кнопки «Відтворити» і «Пауза»

Зліва в іконці ієрархії виберіть тип дрона, потім праворуч з’явиться значок скрипта, де ви можете в реальному часі змінити кількість дронів, а також їх швидкість і висоту (швидкість і висота). . міняти тільки на нові дрони).

Якщо ми натиснемо кнопку «Відтворити» або натиснемо кнопку Build And Run у вікні «Файл», почнеться симуляція і програма запрацює (рис. 5.7).

Рисунок 5.7 – Відображення повітряної обстановки на ІКО

5.5. Висновки за розділом

У розділі було надано опис програмної реалізації та проектування системи керування безпілотними літальними апаратами (БПЛА).

Ця робота включає огляд і аналіз існуючих безпілотних літальних апаратів, а також відомих методів виявлення БПЛА. Розроблено математичні моделі вторгнення дронів різних типів у зону КВОІ, а також параметри польоту дрона – висоту, швидкість, напрямок польоту. За допомогою індикатора кругової зйомки радіолокаційної станції була створена модель динамічного стану погоди в районі дуже важливого інфраструктурного об’єкта. Ця модель включає випадкові потоки повітряних суден, що перетинають зовнішню межу зони, що відповідає об'єкту критичної інфраструктури.

Реалізовано симуляцію польоту двох типів безпілотників, літаків і гелікоптерів. Кожен із цих типів дронів має свою максимальну та мінімальну швидкість, висоту та напрямок. Всі параметри польоту можна повністю редагувати, що забезпечує працездатність і ефективність розробленої програми. Також кількість дронів кожного типу можна редагувати окремо.

6 ЕКОНОМІЧНА ЧАСТИНА

Будь-яка науково-дослідна робота завжди вимагає певних витрат. Для виробництва та реалізації того чи іншого продукту ці витрати необхідно постійно скорочувати, адже це розвиток будь-якого суспільства. Якщо це не так, то будь-яка науково-технічна розробка не буде впроваджена на практиці, оскільки така розробка не буде ефективнішою за наявні на ринку аналоги.

На основі економічних розрахунків можна довести економічну вартість і ефективність застосування результатів науково-дослідної роботи у виробництві, тобто здійснити так звану комерціалізацію наукових розробок.

Дана частина магістерської кваліфікаційної роботи присвячена цим завданням і передбачає наступні етапи роботи (рис. 6.1):

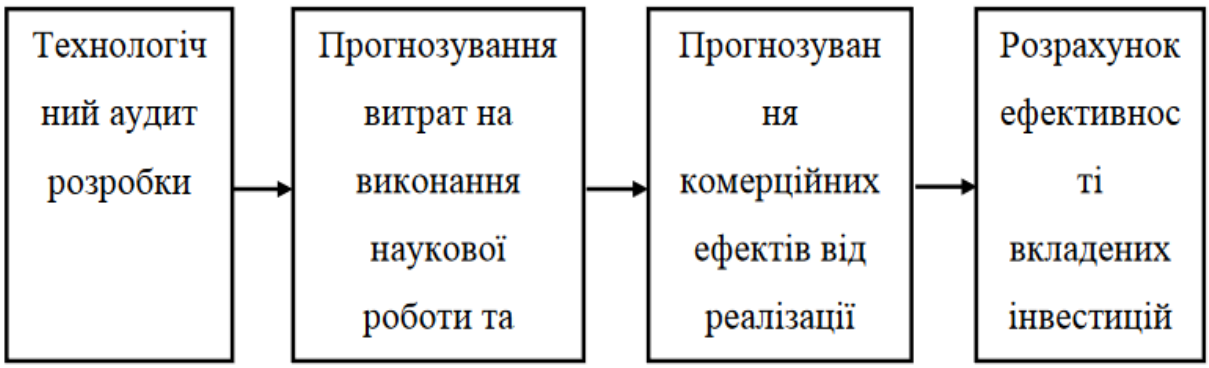


Рисунок 6.1 – Складові економічної частини магістерської кваліфікаційної роботи

Економічна частина даної магістерської роботи буде поділена на наступні розділи. Усі економічні розрахунки будуть включені до зазначених підрозділів економічного розділу. У комплексі ці кроки дозволяють побачити повну картину доцільності розробки та впровадження запропонованого рішення.

6.1. Оцінювання комерційного потенціалу розробки

Оцінка окремо зосереджена на доцільності впровадження нових технологічних ідей та визначенні їх доцільності в промислових масштабах. Такі оцінки зазвичай передбачають розгляд кількох блоків питань, серед яких необхідно вивчити: переваги для споживачів; характеристики потенційного ринку; основні конкуренти; здійсненність ідеї; Постачання ресурсів.

Якщо запропонований продукт стане успішним, усі конкуренти або просто інші підприємці захочуть приєднатися до успіху та виробляти подібні продукти або використовувати ту саму технологію. Тому надійний захист інтелектуальної власності, яка є основою розробки, є важливим фактором зниження ризику передчасного припинення циклу реалізації нового продукту.

Відомі не тільки якісні, а й кількісні методи оцінки комерційного потенціалу технологій, які особливо корисні при проведенні порівняльного аналізу технологій і ранжуванні їх за комерційним потенціалом або відносними ризиками. Для оцінки комерційного потенціалу системи були використані критерії, наведені в таблиці 6.1.

Таким чином, кожній із характеристик присвоюється певний максимальний бал і дається конкретна оцінка цього проекту. Після ідентифікації всіх ознак «вагу» цієї ознаки або цілої групи факторів (наприклад, визначення рівня технологічних переваг) можна розглядати в сукупності загальнозрозумілих параметрів. Для більшої точності оцінки комерційного потенціалу оцінку можуть проводити паралельно декілька експертів у цій галузі, після чого їх бали додаються і виходить усереднена математична оцінка комерційного потенціалу.

6.2. Трудомісткість виконання роботи

Для визначення трудомісткості роботи складається перелік усіх основних етапів роботи.

Особливу увагу приділено логічному розташуванню окремих видів робіт та визначенню можливостей їх паралельного виконання, що дозволяє значно скоротити загальну тривалість роботи.

Таблиця 6.1 – Розподіл робіт по етапах і видам, оцінка їх трудомісткості

№	Етап розробки	Вид роботи на даному етапі	Трудомісткість виконання роботи, люд. х год.
1	Підготовчий	1. Постановка задачі	12
		2. Збір матеріалу та аналіз існуючих розробок	24
2	Постановочний	1. Тестування Unity	8
		2. Тестування Visual Studio	8
		3. Розбір мови програмування C#	12
3	Конструкторська	Створення експериментальної моделі	16
4	Коригуючий	Підготовка попередніх висновків	16
5	Заключення	1. Підведення підсумків	8
		2. Оформлення результатів	16
6	РАЗОМ трудомісткість виконання дипломної роботи		120

Кількість годин активної роботи над проектом відносно дослідження складає 120 (сто двадцять) годин. Робочий час, виділений на дослідження, на добу дорівнює 8 (вісьми) годинам. Звідси випливає, що термін на виконання даного проекту дорівнює 15 (п’ятнадцяти) добам.

6.3. Розрахунок витрат на виконання роботи

Даний проект передбачає розробку інформаційної системи для визначення маршруту польоту БпЛА. Отже для досягнення запланованих цілей, необхідно підрахувати витрати на тестування програми, вартість підсумкової проекту, закупівлю відповідного обладнання, та так само окупність витрат.

На першому етапі дослідження роботи використано наступні технічні засоби:

1) Комп’ютер;

2) Програмне забезпечення;

3) Монітор;

4) Принтер;

Загальна сума витрат на матеріальні ресурси ($З_{мр}$) було розраховано за формулою (6.1).

$$З_{мр} = \sum_{i=1}^{n} Р_i \cdot Ц_i, \qquad (6.1)$$

де $Р_i$ – кількість витраченого і-го виду матеріального ресурсу;

$Ц_i$ – ціна за одиницю і-го виду матеріального ресурсу, грн;

i – вид затраченого матеріального ресурсу;

n – кількість видів затрачених матеріальних ресурсів.

Таблиця 6.2 – Витрати на технічне оснащення

№	Найменування матеріального ресурсу	Одиниця виміру	Кількість використаного матеріалу	Ціна за одиницю, грн	Разом, грн
1	Комп'ютер	Шт.	1	18000	18200
2	Монітор	Шт.	1	1393	1463
3	Принтер	Шт.	1	4376	4673
4	Програмне забезпечення	Шт.	1	24000	24000
5	Підсумкові затрати				48336

Так як для виконання роботи використовується електрообладнання, то проведемо необхідний розрахунок витрат на електроенергію. З допомогою формули (6.2) проводився розрахунок суми загальних витрат на електроенергію.

$$З_{ен} = \sum_{i=1}^{n} М_i \cdot К_i \cdot Т_i \cdot Ц, \quad (6.2)$$

де $М_i$ – паспортна потужність і-х електроприладів, кВт;

$К_i$ – коефіцієнт використання потужності і-х електроприладів;

$Т_i$ – час роботи і-х електроприладів за весь період розробки;

i – вид електроприладів;

n – кількість використаних електроприладів.

Таблиця 6.3 – Витрати на електроенергію

№	Найменування	W, кВт	Коефіцієнт використання, W	Час роботи обладнання для ПП	Ціна, $\frac{\text{грн}}{\text{кВт·год}}$	∑, грн
1	Комп'ютер	0,16	0,8	120	2,64	40.55
2	Монітор	0,05	0,7	120	2,64	11.08
3	Принтер	0,44	0,7	120	2,64	99.57
5	Загалом					151.43

Основна заробітна плата за виконання зазначеної роботи визначається в залежності від ставки тарифної розробника за годину роботи та часу затраченого на кожен етап роботи.

Загальна сума витрат на оплату ($З_{ТР}$) визначається за формулою (6.3).

$$З_{ТР} = \sum_{i=1}^{n} ЧС_i \cdot Т_i, \qquad (6.3)$$

де $ЧС_i$ – часова ставка і-го робітника, грн;

$Т_i$ – трудомісткість розробки ПП, люд.·год.;

i – категорія робітника;

n – кількість робітників.

Розрахунок основної заробітної плати дослідників по даній роботі приведено в табл. 6.4.

Таблиця 6.4 – Витрати на оплату праці

№	Категорія робітника	Кваліфікація	Трудомісткість розробки ПП	Годинникова ставка, грн	∑, грн
1	Науковий керівник	Керівник проекту	10	98	980
2	Розробка документації	Керівник проекту	10	98	980
3	Проектувальник	Розробник	100	50	5000
4	ЗАГАЛОМ				6960

Соціальний внесок становить 22% від фонду оплати праці (ФОП). Пенсійні нарахування не обкладені соціальним податком. Основні засоби розраховуються за допомогою формули (6.4).

$$ОЗ = (ФОП - ПВ) \cdot 22\%, \quad (6.4)$$

де ПВ – пенсійні нарахування, що розраховуються за формулою (6.5).

$$ПВ = ФОП \cdot 10\% = 6960 \cdot 0{,}1 = 696 \text{ грн}, \quad (6.5)$$

$$ОЗ = (ФОП - ПВ) \cdot 22\%, = (6960 - 696) \cdot 0{,}22 = 1378.08 \text{ грн}, \quad (6.6)$$

Загальна сума амортизаційних підраховувань визначається за формулою (6.7).

$$З_{АМ} = \sum_{i=1}^{n} \frac{Ф_i \cdot Н_{А_i} \cdot Т_{р_i}}{100 \cdot Т_{еф_i}}, \quad (6.7)$$

де $Ф_i$ – вартість і-го основного фонду;

$Н_{А_i}$ – річна норма амортизації і-го основного фонду;

$Т_{р_i}$ – час роботи і-го основного фонду за весь період виконання роботи, год;

$Т_{еф_i}$ – ефективний фонд часу роботи і-го основного фонду за рік, л/год;

i – вид основного фонду;

n – кількість основних фондів.

Розрахування амортизаційних відрахувань:

1) Комп’ютер:

$$З_{АМ} = \frac{18200 \cdot 20 \cdot 120}{100 \cdot 2920} = 149.6 \text{ грн};$$

2) Монітор:

$$З_{АМ} = \frac{1473 \cdot 20 \cdot 120}{100 \cdot 2920} = 12.1 \text{ грн};$$

3) Принтер:

$$З_{АМ} = \frac{4673 \cdot 15 \cdot 120}{100 \cdot 2920} = 28.8 \text{ грн};$$

4) Програмне забезпечення:

$$З_{АМ} = \frac{24000 \cdot 15 \cdot 80}{100 \cdot 2920} = 98.6 \text{ грн}.$$

Амортизаційні відрахування приведено в табл. 6.5.

Таблиця 6.5 – Амортизація основних фондів

№	Обладнання	Первісна вартість, грн	Річна норма амортизації, %	Ефективний фонд часу роботи обладнання, л/рік	Час роботи обладнання для виконання роботі, год	Сума, грн
1	Комп’ютер	18200	20	2920	120	149.6
2	Монітор	1473	20	2920	120	12.1
3	Принтер	4673	15	2920	120	28.8
4	Програмне забезпечення	24000	15	2920	80	98.6
5	ПІДСУМОК					289.1

Результати табл. 6.2 – 6.5 приведено в табл. 6.6.

Таблиця 6.6 – Кошторис витрат на виконання роботи

№	Статті витрат	Сума, грн
1	Витрати на матеріали	48336
2	Витрати на оплату праці	6960
3	Ціна на електроенергію	151.8
4	Соціальний податок	1378.08
5	Амортизація основних фондів	289.1
6	ПІДСУМОК	57114.98

Цінцева сума по статті витрат становить 57114.98грн.

6.4. Визначення можливої (договірної) ціни роботи

Можлива (договірна) вартість роботи повинна визначатися на рівні виконання, якості та строків її виконання, що відповідає економічним інтересам користувача (споживача) та оператора.

Договірна ціна ($Ц_Д$) для прикладних робіт вираховується за наведеною формулою (5.7).

$$Ц_Д = З_{роб}\left(1 + \frac{Р}{100}\right), \quad (6.7)$$

де $З_{роб}$ – затрати на виконання роботи (табл. 5.6);

Р – середній рівень рентабельності роботи, % (прийнято 25%);

$$Ц_Д = З_{роб}\left(1 + \frac{Р}{100}\right) = 57114.98\left(1 + \frac{25}{100}\right) = 71393.7 \text{ грн.}$$

Для знаходження ціни реалізації продукту потрібно урахувати податок на додану вартість (ПДВ). Ставки ПДВ 2023, що діють: 20% – загальна ставка пп. "а" п. 193.1 ПКУ[15]. Ціна реалізації розраховується за формулою (5.8).

$$Ц_р = Ц_Д + Ц_Д \cdot ПДВ, \quad (6.8)$$

$$Ц_р = 71393.7 + 71393.7 \cdot 0{,}20 = 85672.44 \text{ грн}$$

Ціна реалізації 85672.44 грн.

6.5. Висновки за розділом

Виконання цього дипломного проекту займає 15 днів. Це включає всі етапи, від роботи до можливості реалізації. Після оголошеного терміну дослідження будуть завершені, всі матеріали підготовлені для подальших досліджень і впровадження технології.

На роботі працюють 2 працівники:

1) керівник проекту, який займається визначенням роботи, визначенням системних вимог, підбором обладнання, інструментів, а також розробкою робочої документації;

2) студент, якому необхідно зібрати матеріал, проаналізувати сучасні розробки, провести безпосереднє дослідження; У результаті вивчено існуючі конструкції ПС СУ та напрацьовано матеріал для подальших досліджень.

Що стосується сфери застосування, то це дослідження сприятиме створенню процедурної системи управління для створення структури через їх низькі вимоги. Крім того, розраховується економічна характеристика роботи. Зазначимо, що вартість даної роботи становить: 71393,7 грн. У підсумку вартість розробки цього проекту становить 85 672,44 грн.

ЗАКЛЮЧЕННЯ

Під час цього дипломного проекту був розроблений модуль керування БПЛА, який дозволяє ігнорувати перешкоди на траєкторії польоту. Крім процесу самоконтролю, сама модель має інтерактивний графічний інтерфейс і може візуально відображати політ БПЛА по заданій траєкторії.

Також були розглянуті предметні області, пов’язані з моделлю. Проаналізовано програмний аналог для керування рухом візуальних об’єктів на екрані, виділено їх переваги та недоліки, які враховано при проектуванні та подальшій розробці моделі.

Методом виконання робіт з математичного моделювання є розробка математичних моделей, реалізація моделей на мові програмування C# в середовищі розробки Unity. Unity надає широкі можливості для створення моделей різних типів для різних сфер використання. Середовище програмування – Visual Studio.

Ця робота включає огляд і аналіз існуючих безпілотних літальних апаратів, а також відомих методів виявлення БПЛА. Розроблено математичні моделі вторгнення дронів різних типів у зону КВОІ, а також параметри польоту дрона – висоту, швидкість, напрямок польоту. За допомогою індикатора кругової зйомки радіолокаційної станції була створена модель динамічного стану погоди в районі дуже важливого інфраструктурного об’єкта.

Отримані в роботі результати можуть бути використані на практиці для створення набору таблиць, пов’язаних з різними умовами та режимами інтенсивності динамічних погодних умов регіону КВОІ, які можуть використовується для моделювання погодних ситуацій різного ступеня складності.

Модуль розроблено на мові програмування C++ з використанням додаткової мультимедійної бібліотеки SFML. Ця бібліотека була використана для графічної частини моделі, що дозволило зробити розробку

більш ефективною, оскільки SFML надала всі необхідні набори для візуалізації роботи моделі.

Під час роботи над створенням дипломного проекту набуваються нові корисні навички:

- Створення програмного коду мовою C++
- Робота з мультимедійною бібліотекою SFML
- Розумна робота з об'єктами в 2D просторі

Також було отримано унікальний досвід створення дипломного проекту за всіма загальноприйнятими стандартами.

Галузь використання отриманих результатів – створення автономних шляхів пересування безпілотних літальних апаратів для проведення зйомки в будь-яких умовах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Чернишев М. І., Куценко В. В. Оцінка точності визначення положення БпЛА різницево-далекомірним методом в рухомій системі пасивної радіолокації зенітних комплексів малої дальності. – Харків: Харків. нац. ун-т Повіт. Сил ім. І. Кожедуба, 2018. – 66с.

2. Синеглазов, В. М., Захарін, Ф. М. Теоретичні основи проектування інтегрованих навігаційних комплексів безпілотних літальних апаратів. – Київ: Освіта України. 2015. 340 с.

3. C++ Documentation. Мова програмування C++ URL: https://devdocs.io/cpp/.

4. Гумен О. М. Технологія автоматизованого геометричного моделювання проективних n-просторів. Харків: Нац. техн. ун-т Укр. «Київ. політх. ін-т». - 2012. - С. 92-96.

5. Шульц Р. В. До питання розрахунку точності визначення координат точок під час аерофотознімання з безпілотних літальних апаратів / Р. В. Шульц, С. П. Войтенко, П. Д. Крельштейн, І. А. Маліна // Інженерна геодезія: наук.-техн. зб. / Київ. нац. ун-т буд-ва та архітектури ; відп. ред. С. Войтенко. - Київ, 2015. - Вип. 62. - С. 124-136.

6. Qt Group. Qt Creator посібник. URL: https://doc.qt.io/qtcreator/.

7. Zelnio A.M. Detection of small aircraft using an acoustic array. Thesis. B.S. /A.M. Zelnio. – Electrical Engineering, Wright State University. - 2007. – 55 p.

8. Bisio I., Garibotto C., Lavagetto F., Sciarrone A., Zappatore S. Blind Detection: Advanced Techniques for WiFi-Based Drone Surveillance. IEEE Trans. Veh. Technol. 2019;68:938–946. doi: 10.1109/TVT.2018.2884767.

9. Gumen O. Research of thermal processes in industrial premises with energy-saving technologies of heating / O. Gumen, N. Spodyniuk, M. Ulewicz, Ye. Martyn // Diagnostyka. – 2017. – Vol. 18. – № 2. – P. 43-49.

10. Пілотажно-навігаційні комплекси [Текст] : конспект лекцій / А. М. Субота, В. Г. Джулгаков, Д. В. Сокол. – Харків : Нац. аерокосм. ун-т ім. М. Є. Жуковського «Харків. авіац. ін-т», 2021. – 108 с.

11. Радзівілов Г. Д., Фесенко О. Д. Аналіз способів реалізації автономних систем навігації БпЛА //Збірник наукових праць [Військового інституту телекомунікацій та інформатизації]. – 2019. – №. 1. – С. 75-81.

12. Басова, А. Є. Методи синтезу систем автоматичної стабілізації та позиціонування [Текст]: навч. посібник / А. Є. Басова, А. С. Кулік, С. М. Пасічник, Н. М. Харіна. - Харків: Нац. аерокосм. ун-т ім. М. Є. Жуковського «Харків. авіац. ін-т», 2019. - 192 с.

13. Колобородов В. Г. Застосування методів і алгоритмів цифрової обробки зображень в оптико–електронних приладах / В. Г. Колобородов, К. В. Харитоненко// Вісник НТУУ "КПІ". – К. : НТУУ «КПІ», 2010. – № 40. – С. 23–31.

14. Субота, А. М. Науково-дослідна робота магістрів [Текст] : навч. посіб. до практ. занять / А. М. Субота, В. Г. Джулгаков. – Харків : Нац. аерокосм. ун-т ім. М. Є. Жуковського «Харків. авіац. ін-т», 2020. – 112 с

15. . Koryttsev, S. Sheiko, V. Kartashov, O. Zubkov, V. Oleynikov, I. Selieznov, M. Anohin. Practical Aspects of Range Determination and Tracking of Small Drones by Their Video Observation // 2020 International Scientific-Practical Confer-ence. Problems of Infocommunications. Science and Technology. Kharkiv, Ukraine. October 6-9, 2020. – 5 p.

16. М.А. Беляева. Моделювання систем. 2012, С. 43-50. URL: http://simulation.su/uploads/files/default/2012-belyaeva-lekcii-part1.pdf.

17. Вірченко С. Г. Деякі питання комплексного динамічного формоутворення на прикладі проектування крила літака / С. Г. Вірченко // Прикладна геометрія та інженерна графіка. - 2018. - Вип. 94. - С. 20-25.

18. V. Kartashov, V. Oleynikov, O. Zubkov, S. Sheiko. Optical detection of unmanned air vehicles on a video stream in a real-time // The Fourth International Conference on Information and Telecommunication Technologies and Radio Electron-ics (UkrMiCo'2019), 9–13 September 2019, Odessa, Ukraine, 4 p.

19. V. Kartashov, V. Oleynikov , I. Koryttsev, S. Sheiko, O. Zubkov, S. Babkin. Processing of Wide Band Acoustic Signals During Detection of Unmanned Aerial Vehicles // 2020 IEEE Ukrainian Microwave Week (UkrMW). Kharkiv, Ukraine,

September 21 - 25, 2020. Volume 1 on 2020 IEEE 12th International Conference on Antenna Theory and Techniques (ICATT). pp. 35-39.

20. Станкевич С. А. Застосування сучасних технологій аерокосмічного знімання в аграрній сфері / Станкевич С. А., Васько А. В. // Наукові аспекти геодезії, землеустрою та інформаційних технологій: матер. наук.-практ. конфер. – 2011. – С. 44–50.

21. Watabe Y., Sassa S. An Effective Investigation of Tidal Flat Sedimentation by Means of UAV and MASW. Journal of Japan Society of Civil Engineers, Ser. B2 (Coastal Engineering), 65(1). – 2009. – P. 1441–1445.

Додаток А

Programiz
C# Online Compiler

Main.cs

```
function () {
  mas = { x: [], y: [] };
  var p=0,vhodflag=false;
  for (let i = 1; i < poligon.length; i++) {
    vhodflag=false;
    for (let j = 0; j < poligon[i].xspare.length; j++) {
      if (inPolyc(poligon[i].xspare[j], poligon[i].yspare[j]) == true)
        vhodflag = true;
    }

  if (vhodflag != false) poligon[i].vhod = true;


    if (poligon[i].vhod==true) {
      p+=Math.abs(Math.round(plosha(poligon[i])))
    }


  }
  document.getElementById("Pa").innerHTML = Math.round(plosha(poligon[0]));
  document.getElementById("Ca").innerHTML = p
});

var mas = { x: [], y: [] };
var lineskal = { x: [], y: [] };
var poligplosh = { x: [], y: [] };
var pocrutie = () => {
  var n = 0,
    flag = false;
  poligon[0].xspare = poligon[0].x;
  poligon[0].yspare = poligon[0].y;
  for (let i = 0; i < 500; i++) {
    if (flag == false) {
      var x2 = poligon[0].x[1];
      var y2 = poligon[0].y[1];
      if (i == 0) {
        var x1 = poligon[0].x[poligon[0].x.length - 1];
        var y1 = poligon[0].y[poligon[0].x.length - 1];
      } else {
        var x1 = poligon[0].x[i - 1];
        var y1 = poligon[0].y[i - 1];
```

```
    } else {
      var x1 = poligon[0].x[1 - 1];
      var y1 = poligon[0].y[1 - 1];
    }
  }
  if (inPolycs(x1, y1) == true) {
    if (poligon[0].x[1 + 1]) {
      console.log("voshol");
      var x2 = poligon[0].x[1 + 1];
      var y2 = poligon[0].y[1 + 1];
    }
    console.log(x1,y1);
    n++;
  } else {
    if (!mas.x.includes(x1)) {
      mas.x.push(x1);
      mas.y.push(y1);
      1--;
    }
    peres = [];
    for (let g = 0; g < poligon.length; g++) {
      if (poligon[g].vhod == true || g == 0)
        for (let h = 0; h < poligon[g].xspare.length; h++) {
          var x3 = poligon[g].xspare[h];
          var y3 = poligon[g].yspare[h];
          if (g == 0 && x3 !== x1 && x3 !== x2) {
            flag = true;
            peres.push({
              x: x3,
              y: y3,
              ugol:
                Math.round(vectorscalyr(x1, y1, x2, y2, x3, y3)
              ),
              dist: disance(x2, y2, x3, y3),
            });
          } else {
            if (
              (Math.round(disance(x1, y1, x2, y2)) ==
                Math.round(disance(x2, y2, x3, y3)) +
                  Math.round(disance(x3, y3, x1, y1)) ||
                Math.round(disance(x1, y1, x2, y2)) ==
                  Math.round(disance(x2, y2, x3, y3)) +
```

```
                Math.round(disance(x2, y2, x3, y3)) +
                  Math.round(disance(x3, y3, x1, y1)) +
                  1 ||
                Math.round(disance(x1, y1, x2, y2)) ==
                  Math.round(disance(x2, y2, x3, y3)) +
                  Math.round(disance(x3, y3, x1, y1)) +
                  -1) &&
              x3 !== x1 &&
              x3 !== x2
            ) {
              flag = true;
              peres.push({
                x: x3,
                y: y3,
                ugol:
                  Math.round(vectorscalyr(x1, y1, x2, y2, x3, y3)
                ), //(x1, y1, x2, y2, x3, y3)
                dist: disance(x2, y2, x3, y3),
              });
            } else if (inPolyc(x3, y3) == true && x3 !== x1 && x3 !== x2) {
              flag = true;
              peres.push({
                x: x3,
                y: y3,
                ugol:
                  Math.round(vectorscalyr(x1, y1, x2, y2, x3, y3)
                ),
                dist: disance(x2, y2, x3, y3), //угол
              });
            }
          }

        sortByugol(peres);
      }
      if (peres.length > 2) {
        console.log(peres);
        for (var d = 0; d < peres.length; d++) {

          if (peres[d].ugol >= -180) {
          }
          /* if (peres[d].ugol == -0) {
```

```
        /* if (peres[d].ugol == -0) {
          peres[d].ugol =0;
        }*/
        if (peres[d].ugol != peres[0].ugol) {
         peres.splice(d, 1);
        }
      }
    }
  }if (peres.length >1) {
    sortByAge(peres);
  }
    if (flag) {
      x1 = x2;
      y1 = y2;
      x2 = peres[0].x;
      y2 = peres[0].y;
      //1--;
      if (x1 == mas.x[0]) {
        return;
      }
    }
  }
  console.log(n);
  poligon[0].xspare = [];
  poligon[0].yspare = [];
};

function sortByugol(arr) {
  arr.sort((a, b) => (a.ugol > b.ugol ? 1 : 0));
}
var disance = (x1, y1, x2, y2) => {
  return Math.sqrt((x2 - x1) ** 2 + (y2 - y1) ** 2);
};

var ochistka = () => {
  for (let l = 0; l < canvas._objects.length; l++) {
    if (canvas._objects[l].radius) {
      canvas.remove(canvas._objects[l]);
      l = 0;
    }
```

```
    }
  }
};

function inPolyc(x, y) {
  var xp = poligon[0].x;
  var yp = poligon[0].y;
  j = poligon[0].LeNgth - 1;
  var c = 0;
  for (var i = 0; i < poligon[0].LeNgth; i++) {
    if (
      ((yp[i] <= y && y < yp[j]) || (yp[j] <= y && y < yp[i])) &&
      x >= ((xp[j] - xp[i]) * (y - yp[i])) / (yp[j] - yp[i]) + xp[i]
    ) {
      c = !c;
    }
    j = i;
  }
  return c;
}

function inPolycs(x, y) {
  var c = 0;
  for (let u = 1; u < poligon.length; u++) {
    var xp = poligon[u].xspare;
    var yp = poligon[u].yspare;
    j = poligon[u].xspare.length - 1;

    for (var i = 0; i < poligon[u].xspare.length; i++) {
      if (
        ((yp[i] <= y && y < yp[j]) || (yp[j] <= y && y < yp[i])) &&
        x >= ((xp[j] - xp[i]) * (y - yp[i])) / (yp[j] - yp[i]) + xp[i]
      ) {
        c = !c;
      }
      j = i;
    }
  }
  return c;
}
```

```
      if (
        ((yp[i] <= y && y < yp[j]) || (yp[j] <= y && y < yp[i])) &&
        x >= ((xp[j] - xp[i]) * (y - yp[i])) / (yp[j] - yp[i]) + xp[i]
      ) {
        c = !c;
      }
      j = i;
    }
  }
  return c;
}

var getVector = function (p1, p2) {
  return {
    x: p2.x - p1.x,
    y: p2.y - p1.y,
  };
};

var dotProduct = function (v1, v2) {
  return v1.x * v2.x + v1.y * v2.y;
};

var crossProduct = function (v1, v2) {
  return v1.x * v2.y - v1.y * v2.x;
};

var getAngle = function (v1, v2) {
  var dot = dotProduct(v1, v2);
  var cross = crossProduct(v1, v2);

  return Math.atan2(cross, dot);
};
var vectorscalyr = (x1, y1, x2, y2, x3, y3) => {
  var v1 = getVector({ x: x2, y: y2 }, { x: x1, y: y1 });
  var v2 = getVector({ x: x2, y: y2 }, { x: x3, y: y3 });
  var alpha = getAngle(v1, v2);
  return (alpha * 180) / Math.PI;
};

```